
ENDLESS
Legend

ENDLESS Legend

Modding Tutorial

Revision 5

Introduction

Welcome to Endless Legend!

This tutorial provides the first and very basic instructions to help you mod the game. More detailed information should be added in the upcoming months.

Otherwise, if you have further questions, please visit our forums:

http://forums.amplitude-studios.com/Modding

Multiplayer games

Please take notice that if you want to play a multiplayer game with a specific mod you will have to ensure that every player have activated it through the main menu.

How to create a mod

Here is a list of information to learn how to mod some parts of the game:

Creation of the modding folder

1) Create a new folder and name it “Modding”.

2) Copy the elements of the Public folder you want to work on.
Location of the public files: <Steam-Library>\SteamApps\common\Endless Legend

4) Paste these elements inside your new folder "Modding" (you can rename it to whatever you want to call your mod).

5) In order to change or to add something to the game you will have to change the xml files inside your mod.

6) As soon as you have made your changes on whatever xml which is available through the public folder, please follow the instructions below.

Details on the specific modding file

In order to run, a new xml file will have to be added at the root of your mod.

This xml file should be named after your mod.
If you called the folder NewSuperMod then this xml will be named NewSuperMod.xml.

When it is created you must add info in it about the things you changed.

For instance: here I dedicated my changes to some elements in GUI and in Localization.

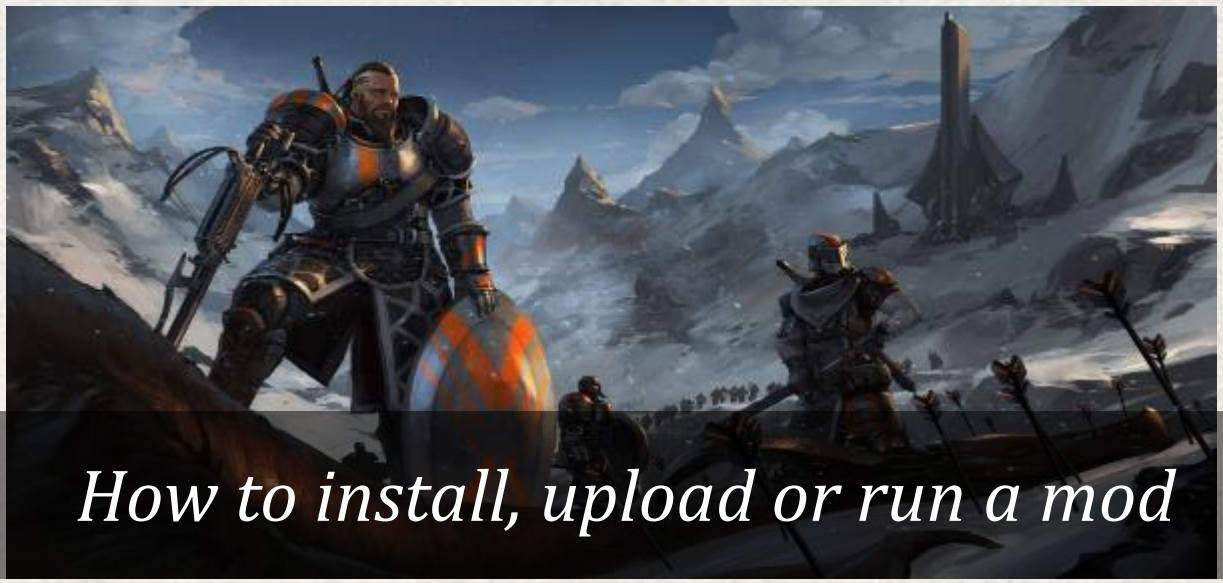
```
<?xml version="1.0" encoding="utf-8" ?>
<Datatable xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <RuntimeModule Name="NewSuperMod" Type="Conversion">
```

```
<Tags />
   <Author>Unknown</Author>
   <Homepage></Homepage>
   <Title>A new super mod!</Title>
   <Version Major="1" Minor="0" Revision="0" />
   <Description>A major new super change to the game.</Description>
   <ReleaseNotes />

   <Plugins>

    <!– Gui –>
    <DatabasePlugin DataType="Amplitude.Unity.Gui.GuiElement, Assembly-CSharp-firstpass">
     <ExtraTypes>
      <ExtraType DataType="Amplitude.Unity.Gui.ExtendedGuiElement, Assembly-CSharp-
firstpass" />
      <ExtraType DataType="QuestGuiElement, Assembly-CSharp" />
      <ExtraType DataType="TechnologyGuiElement, Assembly-CSharp" />
      <ExtraType DataType="TechnologyEraGuiElement, Assembly-CSharp" />
      <ExtraType DataType="UnitSkillGuiElement, Assembly-CSharp" />
     </ExtraTypes>
     <FilePath>Gui/GuiElements[*].xml</FilePath> <—this is the path in your modding file–>
    </DatabasePlugin>

    <!– Localization –>
    <LocalizationPlugin>
     <Directory>Localization</Directory>
    </LocalizationPlugin>

   </Plugins>
  </RuntimeModule>
</Datatable>
```

You can now create a Zip file and upload your mod on the internet.

/!\ IMPORTANT /!
Next to the Name="NewSuperMod" you can read a Type="**Conversion**".
It means that all the files you created will change existing information.
There is another Type="**Standalone**" that will replace the files in the game by the new files you created. You have to be very careful with this Type not forgetting any requirements from any connection otherwise it won't launch.
A last Type="**Extension**" will add new content to the already existing files in the game. It means that a player will be able to launch easily several mod of this type.

To ensure each file you may change is correctly set into the root xml, best should be to use the file in reference (ModdingReference.xml) from the tutorial example. Indeed it contains every bit of database plugin and will cover any changes you could make.
<FilePath> is in fact the path of your own document in the modding folder. The names can be different from the example but you have to ensure the link is always correct between the folder name and the name in this "root" xml.



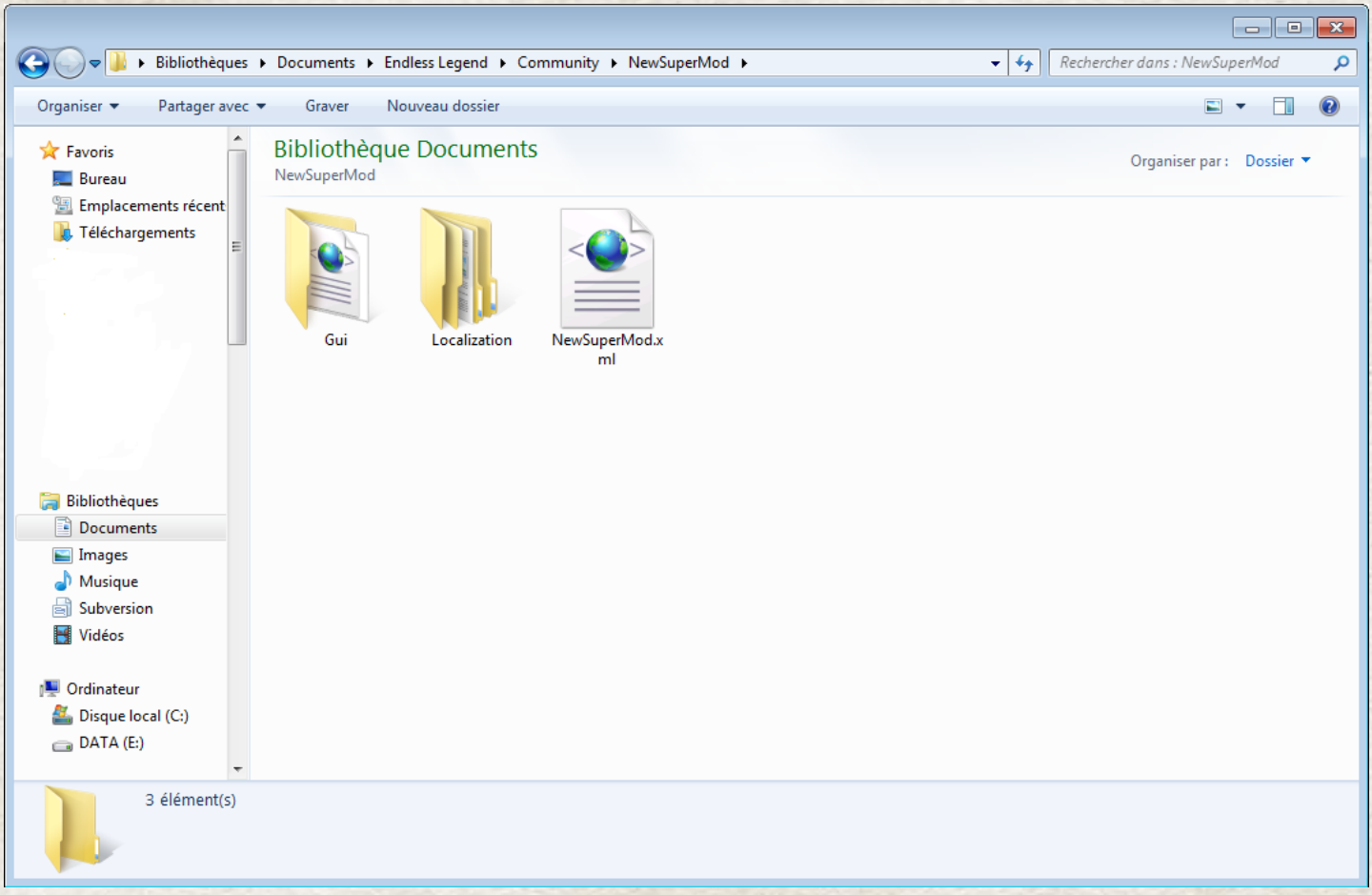
How to install, upload or run a mod

Creating a folder depending your need

1) To make some internal testing:

If not already existing, create a new folder called Community.
On PC: go in my Documents\Endless Legend\Community
On Mac: go in "Library" then open the folder Application support\Endless Legend\Community

Then you need to place in it your modding folder as below (NewSuperMod in example).



2) To upload on Steam:

Right next to the "Community" folder you need to create a "User Generated Content" folder.

Before launching the game you need to go to the properties of your game and insert the launch options: –enablemoddingtools

A watermark is visible on the top right of screen to check this status.

When you go to the new mod screen you will see a "Share" button next to your mod in the list of available mods.

Launching a mod

The in-game mod screen has several sections.

First series of tags determine the type of the mod. You can run multiple mods of extension type but only one conversion or one standalone.

Second series of tags determine the place where are the mod files. On your computer: UGC folder is the new folder from where you will be able to upload your mod on Steam. Community folder is the old one, for private tests and compatibility.

Third series of tags are from the mods themselves. Similar to Steam tags. You can only add those specific tags to your mod.

Here is the detailed list: AI, Art, Buildings, Gameplay, Improvements, MajorFactions, Maps, MinorFactions, Multiplayer, Other, Resources, Technologies, Terrain, Units.

The list of mods shows what is available and you just need to click on them to select them.
Don't be surprised to see a same mod several times in the list. It only means it is located in different places (community + UGC folder most of the time).
You only need to validate your choice to launch your compilation.

At the bottom of the screen you will see a section named
It is because we've implemented a system to save your selection of mods.

- "Save As" button will save the list of mods you chose as a new playlist. A new panel will open in order for you to add some details like picture, title and description.
- "Save" button will only save the changes made on an already existing playlist.

Basics on simulation

In each and every xml of the game you will see text between the marks <PathPrerequisite> or like Path="". What is written follows some rules that allow checking specific elements that are going in your current game.

The structure

The structure is based on hierarchy. There is the Empire at the top and it has a lot of different children which have their own children. It has this kind of look:

ClassEmpire (or EmpireTypeMajor)
- >ClassCity
  - >ClassDistrict
  - >ClassPointOfInterest
- >ClassArmy
  - >ClassUnit
- >ClassEmpirePlan
- >ClassResearch

Then, when you look at a specific moment in-game where one of your army has a hero in it you will make the corresponding path:
../ClassEmpire/ClassArmy/ClassUnit,UnitRankHero

You can see several other info throught this path.

1) The easiest thing to understand is that the / is the link between parent and child.

2) Concerning the points at the beginning.
There are 3 different ways to check elements in the hierarchy.
Let's take an example:

>ClassA
>ClassB
>ClassC

You can write:
"ClassA/ClassB/ClassC" = will mean the check starts at A then ensures B and C are children.
"./ClassC" = will mean that it checks the parent only.
"../ClassB/ClassC" = similar to a folder research. It will look at where is B, whatever place it is, then look if C is a child.
".../ClassA" = will mean it ensures to go at the root of the simulation, whatever the place of A is.

3) Concerning the element after the comma (UnitRankHero in this case).
This is what we call a Simulation Descriptor wich is referenced into the Class. This is why there is only a comma between ClassUnit and UnitRankHero. You could even rely only on a Descriptor and forgot the class in this case.

What you have to remember is that each time you create an object it should have a descriptor to work correctly (simulationdescriptor often contains game effects).
Then, if you want to check for example that your new city improvement is built in a city the path will be:
../ClassCity/CityImprovementNew

4) Last information to take in consideration is that each class has properties that can be also check through calculation.
It is mainly numbers and you can check them through formulas thanks to
<InterpreterPrerequisite > instead of <PathPrerequisite>

For example, in the ClassEmpire (EmpireTypeMajor), you can check several numbers:
AllianceCount, WarCount, HeroCount, EmpirePointStock, CurrentTurn, EmpireApproval, etc.

Simulation hierarchy

The path will always depend on where you are starting in the hierarchy.

If you start on the empire (like it is for the faction traits) and you want to modify a property in all the unit of all the armies of your empire, you have to write this path:
ClassEmpire/ClassArmy/ClassUnit

If you are modifying a unit ability which is applied directly on a unit and you want to modify a property in the corresponding unit your path will be:
ClassUnit or nothing

If you are modifying a unit ability which will modify a property on all units alongside it and this in a city or an army then you will need
../Garrison/ClassUnit

New explanation again on the dots subject:
../ will try you then your parent and the parent of the parent and... until it find the next tag which is "Garrison" in the example. So if you are starting on a unit in an army, the path will ask the unit if it has the "Garrison" tag which is not the case, then ask the parent of the unit. The parent is the army and an army has the tag "Garrison" so the path will say "ok, I can continue the search from the army".
It will then try to find the tag "ClassUnit" in all the army children.

Sealed properties

Another good thing to know concerns the sealed properties.

You will find them sometimes. They are used to define a property which is not modifiable by modifier and simulation. They are used when we need to store a value computed in the code. Like the health of a unit for example.
If you want to modify the Health of a unit you have to modify the MaximumHealth or the unit regeneration but you should never try to modify directly the health.

Also, there is no one shot instruction in the simulation. Like, “loose 10 hp now”.

You should go to the last pages of this tutorial to have a look at the advanced parts about simulation descriptors and Interpreter prerequisite for a bit more information.

How to tweak season

Global Tags

In the simulation you have to take note that the season is always known.
If you want to use this info at your advantage you can use in the path the following mention:
#Winter
#Summer

Adapting/adding winter effects

Ref: Simulation\SeasonEffects[Winter].xml

- Name: The winter effect's "id", serves as a tag to identify it.

- SeasonName: It is only possible to add effect for winter.

- NumberOfPastSeason: It will indicate that the effect could trigger starting from the winter number X. If it is not trigger, it will add itself to the pool of available effects for the next winter. Please note that the first winter is a bit special because all effect set to 1 will be in the list of first winter maluses.

Ref: Simulation\SimulationDescriptors[Season].xml

About the simulation descriptor it should have the same name in both files.
But it is in the xml just above that you will set up the effects itself.
The specifity here is that every of those simulation descriptors are modifiers. It means that they are modifying existing values (don't forget to check winter immunities as in the example below).
Please do not hesitate to have a look at the end of the document to have more info on possibility (Advanced: Simulation Descriptor)

Example:

```
<SimulationDescriptor Name="YourWinterMalus">
  <SimulationModifierDescriptors>
    <SimulationModifierDescriptor TargetProperty="CityFood"       Operation="Percent"
Value="-0.25" Path="EmpireTypeMajor/!WinterFoodReductionImmunity,ClassCity"/>
  </SimulationModifierDescriptors>
</SimulationDescriptor>
```

How to add a new faction trait

Create the GUI element (in game visibility)

Ref: Gui\GuiElements[FactionTraits].xml

- Name: The trait's "id", serves as a tag to identify it in any xml file. Different levels for a same trait should have different names (eg: NewTrait1, NewTrait2, etc.).

- Title: If you want your faction trait to be localized, use"%" to reference a localization key (eg: %CrusaderTitle) and go to a). Otherwise skip to b).

a) Localized text version: Open EF_Localization_Assets_Locales.xml in MyMod\Localization\english (or other languages). Create a new LocalizationPair with the %name (eg: %NewTraitTitle) and the actual name (eg: The new trait). Translate appropriately in the other Localization_Locales files.

b) Simply write the trait's name

- Description: Same as Title.

Create the trait reference

Ref: Simulation\FactionTraits[Custom].xml

- Name: The trait's "id", serves as a tag to identify it in any xml file. Has to be the same as the Gui element.

- SubCategory: There are different categories of trait according to their purpose. "StandardTrait" should be used for new trait.

- Family: To be set as a "Trait" in this case.

- Custom: When you add a trait you must set this value to "true" to make it available for custom faction.

- Cost: Will be the cost of the custom trait.

Create the effects

Ref: Simulation\SimulationDescriptors[FactionTrait].xml

- Name: The trait's "id", serves as a tag to identify it in any xml file. Has to be the same as the Gui element and custom faction trait.

- Type: It is a "FactionTrait" in this case.

- SimulationModifierDescriptors: Those will be the formulas used to create the effects.
It is written as follows:

```
<SimulationModifierDescriptors>
 <SimulationModifierDescriptor TargetProperty="MaximumMovement"  Operation="Addition" Value="2"  Path="EmpireTypeMajor/Garrison/ClassUnit,#Winter"/>
 <SimulationModifierDescriptor TargetProperty="VisionRange"  Operation="Addition" Value="1"  Path="EmpireTypeMajor/Garrison/ClassUnit,#Winter"/>
</SimulationModifierDescriptors>
```

TargetProperty: The variable you want to modify. The target properties accessible can be found in majority in **SimulationDescriptors[Class].xml**.
Operation: It can be a multiplication, addition, percentage, subtraction, division or power.
Value: By how much you want to modify the variable (positive or negative)

Path: Very important. It shows what TargetProperty you want to modify exactly.
Eg: You want to modify Approval for your empire, affecting all the cities then Path=”EmpireTypeMajor/ClassCity”. However, if you want to modify Approval only for a specific affinity, add that affinity using a “,”. Path = “EmpireTypeMajor,AffinityBrokenLords/ClassCity”.
Other eg: You want to change the DamageMax for unit type ranged only. This is the Path you would use: EmpireTypeMajor/ClassArmy/ClassUnit,UnitTypeRanged.

How to use 3D models

You cannot import new 3D models. But you can try to swap a model with another or create one from scratch. Like spawing a battle mage with the broken lords Stalwarts model or create a new faction which has the broken lords Ryder, the Wild Walkers Dekari ranger and a necro Profilerator as base units.

Change the call type of the 3D model

Ref: Mapping\Mapping.xml

In fact, we use the file "Mapping/Mapping.xml" to define which 3d asset to spawn based on the simulation and some other variables. So if you open it and if you go down a little in the file you should find something like: "WorldPawn". And you can see that we use the "BodyDefinitionName" for looking through the 3d asset.
Change it to a “CustomMapping” descriptor as described below:

```
 <XmlMapping Name="WorldPawn">
  <GameObject Name="$Descriptor(UnitType)"
Prefab="Prefabs/Armies/$Descriptor(CustomMapping)$Descriptor(:Variation)"/>
  <Fallback>
   <GameObject Name="$Descriptor(UnitType)"
Prefab="Prefabs/Armies/$(BodyDefinitionName)$Descriptor(:Variation)"/>
   <Fallback>
    <GameObject Name="$Descriptor(UnitType)"
Prefab="Prefabs/Armies/$(BodyDefinitionName)"/>
    <Fallback>
     <GameObject Name="DefaultPawn" Prefab="#Default World
Mapping/#DefaultPawnPrefab"/>
    </Fallback>
   </Fallback>
  </Fallback>
 </XmlMapping>
```

Create a new descriptor

Ref: Simulation\SimulationDescriptors[UnitType].xml

The name of the descriptor is really important to swap a unit model or to create a new unit! Prefix it by the descriptor type and then the unit body name that you target. When you will try to map the asset 3D, the engine should remove the descriptor type from the name of the descriptor and search for the correct body name without any additional code.

```
<SimulationDescriptor Name="CustomMappingUnitBodyMadFairiesForestSpirit"
Type="CustomMapping"/>
```

Thus it will call the 3D model of the Tenei Walker from Wild Walkers faction.

It is in the part that you can also add the base attributes in order to balance your unit:

```
<SimulationModifierDescriptors>
  <SimulationModifierDescriptor  TargetProperty="BaseAttributeAttack"
Operation="Addition"  Value="18"/>
  <SimulationModifierDescriptor  TargetProperty="BaseAttributeDefense"
Operation="Addition"  Value="46"/>
  <SimulationModifierDescriptor  TargetProperty="BaseAttributeInitiative"
Operation="Addition"  Value="10"/>
  <SimulationModifierDescriptor  TargetProperty="BaseAttributeDamage"
Operation="Addition"  Value="22"/>
  <SimulationModifierDescriptor  TargetProperty="BaseMaximumHealth"
Operation="Addition"  Value="140"/>
</SimulationModifierDescriptors>
```

Change an existing unit with a dedicated model

Ref: Simulation\ UnitBodyDefinitions.xml

Once you choose the UnitBody you want to replace with a new 3D model, lets take the broken lords infantry as an example, you must change the line:
<SimulationDescriptorReference Name="UnitTypeBrokenLordsInfantry" />
And put this as replacement:
<SimulationDescriptorReference Name="CustomMappingUnitBodyMadFairiesForestSpirit"/>

Considering the change of model you must adapt some elements like the number of units on a tile: <SimulationDescriptorReference Name="UnitSizeLarge" />
Or the type of weapons it can carry (depending on the model):
<SimulationDescriptorReference Name="ItemSlotAxe2" />
<SimulationDescriptorReference Name="ItemSlotClaw2" />

Here is the list of requirements.First is the faction name. Then in-game name of unit (underlined). Then xml name, type and details on weapon(s) available (animations).

BROKEN LORDS

Stalwarts = Infantry, melee: Sword1, Shield, Hammer1
Ryder = Cavalry, cavalry: Shield, Hammer1, Spear2
Dust Bishop = BlackBishop, support: Wand2, Scepter2

ARDENT MAGES

Telsem Warlock = BattleMage, melee: Spear2, Claw2
Ateshi Zealot = Witch, support: Staff2, Wand2
Eneqa Wing = Phoenix, flying: Claw2

WILDWALKERS

Dekari Ranger = Archer, distance: Bow2, Crossbow2
Agache Shaman = Shaman, support: Staff2, Wand2
Tenei Walker = ForestSpirit, melee: Claw2, Hammer2

NECROPHAGES

Forager = Necrohunter, melee: Axe1, Shield, Claw2
Necrodrone = Zombat, flying: Claw2
Proliferator = Lichbug, support: Claw2, Wand2

ROVING CLANS

Dervish = Dervish, cavalry: Spear2, Hammer1, Shield
Kassai = Kassai, distance: Bow2, Crossbow2
Yirmak = Berzerk, melee: Spear2, Claw2

DRAKKENS

Drakkenling = Drakkenling, melee: Sword1, Shield, Spear2
Wyvern = Wyvern, flying: Claw2
Ancient = AncientDragon, support: Claw2, Wand2

VAULTERS

Titan = Titan, melee: Sword1, Shield, Claw2
Marine = Marine, distance: Bow2, Crossbow2
Dawn Officer = DawnOfficer, cavalry: Axe1, Shield, Axe2

MINOR FACTION

Rumbler = UrcesGiantOgre, melee: Hammer2, Claw2
Vinesnake = ErycisHydra, melee: Claw2
Drider = CeratanDrider, support: Wand2, Claw2
Daemon = KazanjiDevil, flying: Axe2, Claw2
Centaur = BosCentaur, cavalry: Axe1, Shield, Spear2
Harmonite = SilicsGolem, melee: Hammer2, Claw2
Dredge = DelversDwarf, melee: Hammer1, Shield, Hammer2
Ended = HauntsSpecter, flying: Sword2, Spear2
Justicere = SistersOfMercyJusticere, support: Sword1, Shield, Spear2
Tetike = JotusEttin, distance: Bow2, Crossbow2
Orc = HurnasOrc, distance: Bow2, Crossbow2
Minotaur = GauranMinotaur, cavalry: Axe2, Spear2
Arpuja = BirdhiveHarpy, flying: Axe1, Shield, Spear2
Ice Warg = GeldirusIceWarg, cavalry: Claw2
Caecator = EyelessOnesCaecator, support: Staff2, Scepter2

Change the Gui Element

Ref: Gui\GuiElements[BodyDefinitions].xml AND **Gui\GuiElements[UnitDesigns].xml**

Of course, you should not forget to change the GuiElements with the correct text and image according to the correct [UnitBody] or [UnitDesign] used.
Follow the instructions of next chapters about changing localization and assets to do so.

Add a new unit with a dedicated model

Ref: Simulation\UnitBodyDefinitions.xml

If you want to make a new unit you will have to create a new UnitBody with its own characteristics. Do not hesitate to look at existing template.

```
<UnitBodyDefinition Name="UnitBodyNewUnit" SubCategory="SubCategoryInfantry">
```

Let's begin with the unit abilities:

```
<UnitAbilityReference Name="UnitAbility1"/>
<UnitAbilityReference Name="UnitAbility2"/>
```

Let's continue with the prerequisite (you can unlock it through a technology or give at start of game and dedicate it to a precise faction)

```
<PathPrerequisite
Flags="Prerequisite">.../EmpireTypeMajor,AffinityFaction1</PathPrerequisite>
<TechnologyPrerequisite
Flags="Prerequisite,Discard,Technology">TechnologyFaction1Unit2</TechnologyPrerequisite>
```

The cost:

```
<Cost ResourceName="Strategic4" Instant="true">30</Cost>
<Cost ResourceName="Production" Instant="false">2000</Cost>
```

Here come the simulation descriptors references. Remember the new descriptor you created? Among the other thing you will have to use it here instead of basic "UnitType" thing:

```
<SimulationDescriptorReference Name=" CustomMappingUnitBodyMadFairiesForestSpirit " />
```

Considering the change of model you must adapt some elements like the number of units on a tile: `<SimulationDescriptorReference Name="UnitSizeLarge" />`
Or the type of weapons it can carry, depending on the model you use (cf previous list):

```
<SimulationDescriptorReference Name="ItemSlotAxe2" />
<SimulationDescriptorReference Name="ItemSlotClaw2" />
```

Then the remaining part concern UnitBodyStance and UnitBodyBattleParameters. Once again best is to use a template.

Ref: Simulation\ UnitDesigns.xml

There comes the time to create the unit design itself.

```
<UnitDesign Name="UnitDesignNewUnit#1"          Model="0013234">
<UnitBodyDefinitionReference Name=" UnitBodyNewUnit" />
</UnitDesign>
```

Be very careful to use a model number not already existing.
Do not forget to add its Gui element too.

Ref: Simulation\ FactionTraits[Major].xml AND **Simulation\ FactionTraits[Affinity].xml**

To make a proper link of the unit to a faction you will have to create a faction trait.

A major one first.

```
<FactionTrait Name="FactionTraitNewUnit1"       SubCategory="UnitTrait">
  <FactionTraitTooltipOverride Type="Constructible" OverrideName=" UnitBodyNewUnit "/>
</FactionTrait>
```

Then adding it to an existing list, let's say broken lords.

```
<FactionAffinity            Name="AffinityBrokenLords"
  <SubTrait                 Name=" FactionTraitNewUnit1"/>
</FactionAffinity>
```

As usual, do not forget to add its Gui element.

Unit reskin

There is a possibility to work on the unit textures in order to change their look.

A tutorial with files to download as examples is available on our forums:
http://forums.amplitude-studios.com/showthread.php?84570-Unit-Reskin-Tutorial

We encourage you to get all the information there.

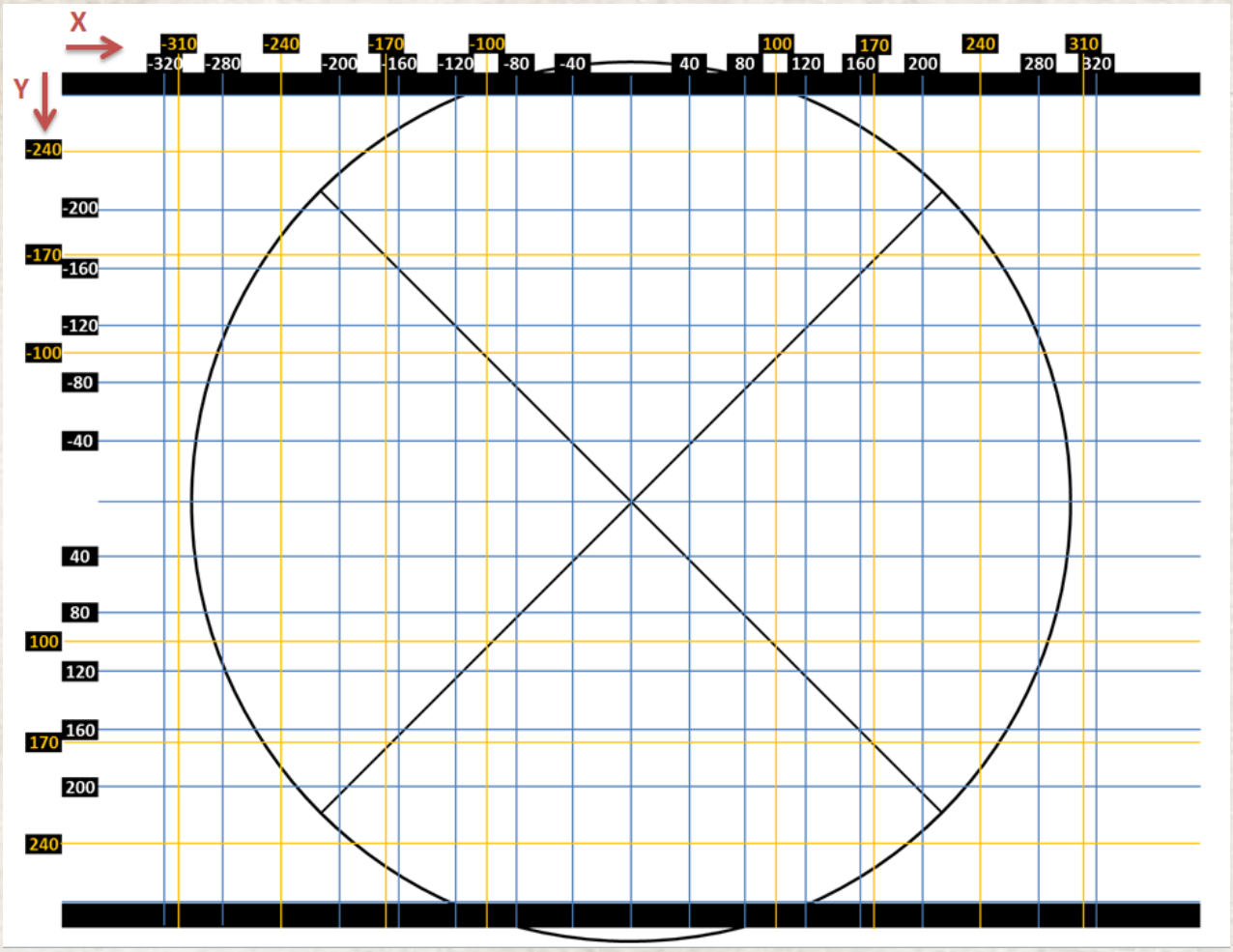
How to add a technology

Create the GUI element (in game visibility)

Ref: Gui\GuiElements[Technology].xml

- Name: The technology definition name, serves as a tag to identify it in any xml file. Each technology should have different names (eg: Technology1, Technology 2, etc.).

- Coordinates: Find below the coordinates inside an era. It works with technology barycentre.



- Title: If you want your technology to be localized, use"%" to reference a localization key (eg: % Technology Title) and go to a). Otherwise skip to b).

a) Localized text version: Open EF_Localization_Assets_Locales.xml in MyMod\Localization\english (or other languages). Create a new LocalizationPair with the %name (eg: %NewTechnologyTitle) and the actual name (eg: The new technology). Translate appropriately in the other Localization_Locales files.

b) Simply write the technology's name

- Description: Same as Title.

- Icon: Please refer to the modding/assets part to learn how to add pictures

- TechnologyEra: Depends in which era you want to see your technology placed, TechnologyEraDefinition1, or 2, or 3, etc.

Create the technology reference

Ref: Simulation\DepartementOfScience+Constructibles[EraXTechnology].xml
(The "X" represents the era the technology will be in)

- Name: The technology definition name, serves as a tag to identify it in any xml file. Has to be the same as the Gui element.

- Category and SubCategory: There are different category and subcategories of technologies according to their purpose. It depends on the major bonus.

- SimulationDescriptorReference: There are 2 different thing. The technology count of the era in which is your technology and the reference to the effects descriptor (see below where to create effects).

- Cost: It is a formula common to each tehcnology depending once again their era.

- PathPrerequisite: The era which is the basic condition to be unlockable.

Create the effects

Ref: Simulation\SimulationDescriptors[Technology].xml

- Name: The technology's "id", serves as a tag to identify it in any xml file. It has to be the name refered into the SimulationDescriptorReference of the technology definition.

- Type: It is a "Technology" in this case.

- SimulationModifierDescriptors: Those will be the formulas used to create the effects. It is written as follows.

```
<SimulationModifierDescriptors>
 <SimulationModifierDescriptor TargetProperty="AttributeAttack" Operation="Addition" Value="30" Path="./EmpireTypeMajor/Garrison/ClassUnit"/>
 <SimulationModifierDescriptor TargetProperty="AttributeDefense" Operation="Percent" Value="0.20" Path="./EmpireTypeMajor/Garrison/ClassUnit"/>
</SimulationModifierDescriptors>
```

TargetProperty: It is the origin of the applied effect.

Operation: It can be a multiplication, addition, percentage, subtraction, division or power.

Value: The number taken into account for the operation

Path: The target of the effect.

Add the new object for the AI

Ref: AI\AIParameters[Technology].xml

For each technology, you must write an AIParameterDatatableElement (as explained in the AI part).

Technologies' data can be converted (with AIParameterConverters) only in these AIParameters:

- AIEmpireGrowth
- AIEmpireProduction
- AIUnitProduction
- AIBuildingProduction
- AIEmpireMoney
- AIEmpireResearch
- AIEmpireEmpirePoint
- AIEmpireDefense
- AIEmpireStrategicResource1
- AIEmpireStrategicResource2
- AIEmpireStrategicResource3
- AIEmpireStrategicResource4
- AIEmpireStrategicResource5
- AIEmpireStrategicResource6
- AIEmpireLuxuryResource
- AIEmpireImproveSearch
- AIEmpireImproveBribe
- AIEmpireImproveTalk
- AIEmpireBuyout
- AIEmpireMilitaryDefense
- AIEmpireMilitaryOffense
- AIEmpireBoosterFood
- AIEmpireBoosterIndustry

- AIEmpireBoosterSciene
- AIEmpireDiplomacy
- AIEmpireMinorFactionAssimilation
- AIEmpireBoosterScience
- AIEmpireExploration

How to add a city improvement

Create the GUI element (in game visibility)

Ref: Gui\GuiElements[CityImprovement].xml

- Name: The city improvement's "id", serves as a tag to identify it in any xml file. Each city improvement should have different names (eg: CityImptovement1, CityImptovement2, etc.).

- Title: If you want your city improvement to be localized, use"%" to reference a localization key (eg: %BuildingTitle) and go to a). Otherwise skip to b).

a) Localized text version: Open EF_Localization_Assets_Locales.xml in MyMod\Localization\english (or other languages). Create a new LocalizationPair with the %name (eg: %NewCityImprovementTitle) and the actual name (eg: The new city improvement). Translate appropriately in the other Localization_Locales files.

b) Simply write the building's name

- Description: Same as Title.

- Icon: Please refer to the modding/assets part to learn how to add pictures

Create the city improvement reference

Ref: Simulation\DepartementOfIndustry+Constructibles[CityImprovement].xml

- Name: The city improvement's "id", serves as a tag to identify it in any xml file. Has to be the same as the Gui element.

- SubCategory: There are different categories of city improvement according to their purpose. It depends on the major bonus.

- SimulationDescriptorReference: This a reference to the effects descriptor (see below where to create effects).

- Cost: Will be the cost of the city improvement.

- PathPrerequisite: All the conditions required to make the city improvement appear. You can hide it for some faction. Commonly it is done for some factions like the faction tutorial.

```
<PathPrerequisite Inverted="true"
Flags="Prerequisite,Discard">../ClassEmpire,AffinityTutorial</PathPrerequisite>
```

You also have to add a line to check if the city improvement is not already built in the empire or the city.

```
<PathPrerequisite Inverted="true"
Flags="Prerequisite,Discard">.../ClassEmpire/Uniques,CityImprovementNew</PathPrerequisite
>
```

Note: In the example the city improvement is a unique one.

Create the effects

Ref: Simulation\SimulationDescriptors[CityImprovement].xml

- Name: The city improvement's "id", serves as a tag to identify it in any xml file. Has to be the same as the simulation descriptor line you write in the city improvement reference.

- Type: It is a "CityImprovement" in this case.

- SimulationModifierDescriptors: Those will be the formulas used to create the effects.
It is written as follows:

```
<SimulationModifierDescriptors>
<SimulationModifierDescriptor TargetProperty="CityFood" Operation="Addition"
Value="5" Path="./ClassCity"/>
<SimulationModifierDescriptor TargetProperty="CityIndustry" Operation="Percent"
Value="0.15" Path="./ClassCity"/>
<SimulationModifierDescriptor TargetProperty="VisionRange" Operation="Addition"
Value="2" Path="./ClassCity/ClassDistrict"/>
<SimulationModifierDescriptor TargetProperty="CityExpansionDisapproval"
Operation="Multiplication" Value="0" Path="./ClassCity"/>
</SimulationModifierDescriptors>
```

TargetProperty: It is the origin of the applied effect.

Operation: It can be a multiplication, addition, percentage, subtraction, division or power.

Value: The number taken into account for the operation.

Path: The target of the effect.

Add the new object for the AI

Ref: AI\AIParameters[CityImprovement].xml

For each improvement, you must write an AIParameterDatatableElement (as explained in the AI part).

Improvements' data can be converted (with AIParameterConverters) only in these AIParameters:

- AICityGrowth
- AICityEmpirePoint
- AICityProduction
- AICityResearch
- AICityMoney
- AICityMoneyUpkeep
- AICityMilitary
- AIEmpireExploration
- AIEmpireColonization
- AIEmpireMilitaryDefense
- AIEmpireMilitaryOffense
- AICityLuxuryResource
- AICityStrategicResource
- AICityGrowth
- AICityApproval
- AICityProduction
- AICityMoney
- AICityResearch
- AICityEmpirePoint
- AICityLevelUp
- AICityDistrictNeighbours

Unlocking a city improvement through a technology

As you can see in the technology eras, most of the city improvements are unlocked thanks to a technology.
Here is the way to make the link between the 2 elements by adding some files related to creation of a new technology.

Ref: DepartmentOfScience+Constructibles[EraXTechnology].xml
No need for a lot of details.
Just create the technology definition of the technology that will be used.

```
  <TechnologyDefinition  Name="TechnologyDefinitionNewCityImprovement"
TechnologyFlags="Affinity" Visibility="VisibleWhenUnlocked">
    <SimulationDescriptorReference Name="TechnologyNewCityImprovement" />
  </TechnologyDefinition>
```

Ref: Simulation\SimulationDescriptors[Technology].xml
No need for a lot of details either.
Just create an empty technology descriptor. The effects will come from the city improvement itself. Must be the same name as in the previous reference though.

```
  <SimulationDescriptor Name=" TechnologyNewCityImprovement"
Type="Technology" />
```

Ref: Simulation\DepartementOfIndustry+Constructibles[CityImprovement].xml
This is where your city reference should be.
You only need to add a technology prerequisite to make the link with your new technology.

```
<TechnologyPrerequisite
Flags="Prerequisite,Discard,Technology,AIEconomicRatio">TechnologyNewCityImprovement</T
echnologyPrerequisite>
```

Ref: GuiElements[Technology].xml

Of course you must not forget to add a Gui element for the technology. Please look at the details on creation of a technology to know how to do this.

How to add an item

Create the GUI element (in game visibility)

Ref: Gui\GuiElements[ItemDefinitions].xml

- Name: The item's "id", serves as a tag to identify it in any xml file. Each item should have a different name (eg: item 1, item 2, etc.).

- Title: If you want your item to be localized, use"%" to reference a localization key (eg: %ItemTitle) and go to a). Otherwise skip to b).

a) Localized text version: Open EF_Localization_Assets_Locales.xml in MyMod\Localization\english (or other languages). Create a new LocalizationPair with the %name (eg: %NewItemTitle) and the actual name (eg: The new item). Translate appropriately in the other Localization_Locales files.

b) Simply write the item's name

- Description: Same as Title.

- Icon: Please refer to the modding/assets part to learn how to add pictures

Create the item reference

Ref: Simulation\ItemDefinitions[Unique].xml

- Name: The item's "id", serves as a tag to identify it in any xml file. Has to be the same as the Gui element.

- SubCategory: There are different subcategories of item according to their genre (eg: For armors it is legs, torso, head)

- Tags: This is an optional info. It can be used to ensure the item will never move into the obsolete part of the equipment vault.

- ConstructibleElementReferenceToInheritFrom: this is something used a lot for item. It means that if I create a new Head armor, adding the "ItemHeadBase" definition will automatically attached all the simulation descriptors from this object.

```
<ItemDefinition Name="ItemHeadBase" Hidden="true" SubCategory="SubCategoryHead">
   <SimulationDescriptorReference Name="ItemCategoryArmor" />
   <SimulationDescriptorReference Name="ItemClassHead" />
   <Slot Name="Head"/>
   <PathPrerequisite Flags="Prerequisite,Discard"
Inverted="false">../ClassUnit,ItemSlotHead</PathPrerequisite>
 </ItemDefinition>
```

In this case it will take the category, the class, the slot and ensuring this one is not already used. In other words, each time someone creates a Head armor it prevents the creation of clone lines.

- SimulationDescriptorReference: As usual their can be multiple references to simulation descriptors already existing or that you will create.
At least you must mention the tier and the type of material. Then you need to make a link to the simulation descriptors you will create for the effects. Please see next step for more info.

- PathPrerequisite: All the conditions required to make the item appear. You can hide it for some factions for instance or only for heroes.

Create the effects

Ref: Simulation\SimulationDescriptors[ItemUnique].xml

- Name: The item's "id", serves as a tag to identify it in any xml file. Has to be the same as the simulation descriptor line you write in the city improvement reference.

- Type: It is a "ItemAttributes" in this case.

- SimulationModifierDescriptors: Those will be the formulas used to create the effects.
It is written as follows.

```
   <SimulationModifierDescriptors>
    <SimulationModifierDescriptor TargetProperty="EquipmentDefense"      Value="3"
Operation="Addition" Path="../ClassUnit" />
    <SimulationModifierDescriptor TargetProperty="EquipmentMaximumHealth"      Value="3"
Operation="Addition" Path="../ClassUnit" />
   </SimulationModifierDescriptors>
```

TargetProperty: It is the origin of the applied effect.

Operation: It can be a multiplication, addition, percentage, subtraction, division or power.

Value: The number taken into account for the operation.

Path: The target of the effect.

Unlocking an item through a technology or an event

As you can see in the technology eras, some of the items are unlocked thanks to technologies. But you can also reward it through a droplist.
In both cases you will need to make a link between the 2 elements by adding some files related to the creation of a new technology.

Ref: DepartmentOfScience+Constructibles[EraXTechnology].xml
No need for a lot of details.
Just create the technology definition of the technology that will be used.

```
 <TechnologyDefinition  Name="TechnologyDefinitionNewItem"
TechnologyFlags="Affinity Quest" Visibility="VisibleWhenUnlocked">
   <SimulationDescriptorReference Name="TechnologyNewItem" />
 </TechnologyDefinition>
```

If this technology is supposed to stay hidden (if item is a quest reward) put Visibility=”Hidden”

Ref: Simulation\SimulationDescriptors[Technology].xml
No need for a lot of details either.
Just create an empty technology descriptor. The effects will come from the item itself. Must be the same name as in the previous reference though.

```
 <SimulationDescriptor Name=" TechnologyNewItem"    Type="Technology" />
```

Ref: Simulation\ItemDefinitions[Unique].xml
This is where should be your item reference.
The only need is to add a technology prerequisite to make the link with your new technology.

```
      <TechnologyPrerequisite
Flags="Prerequisite,Discard,Technology">TechnologyNewItem</TechnologyPrerequisite>
```

Ref: GuiElements[Technology].xml
Of course you must not forget to add a Gui element for the technology definition. Please look at the details on creation of a technology to know how to do this.

If you want to unlock this item through quest reward you will have to use in the droplist the previous Gui element and the one of the **GuiElements[ItemDefinitions].xml**:

```
 <Blueprint Name="TechnologyDefinitionNewItem"  ConstructibleName="ItemNew"
Weight="1"/>
```

How to add a skill

Create the GUI element (in game visibility)

Ref: Gui\GuiElements[UnitSkill].xml
- Name: The skill's "id", serves as a tag to identify it in any xml file. Each skill should have different names (eg: Skill1, Skill2, etc.).

- Coordinates: The sector is the zone (left=0, center=1, right=2). Radius is the "floor" (0= at the bottom, 1= a level higher, etc.) Angle is the position inside the radius which will depend of the place inside the radius chosen (0= to the left and a greater number= going to the center/right).

- Title: If you want your skill to be localized, use"%" to reference a localization key (eg: %SkillTitle) and go to a). Otherwise skip to b).

a) Localized text version: Open Localization_Locales.xml in MyMod\Localization\english (or other languages). Create a new LocalizationPair with the %name (eg: %SkillTitle) and the actual name (eg: The new skill). Translate appropriately in the other Localization_Locales files.

b) Simply write the skill's name

- Description: Same as Title.

- Icon: Please refer to the modding/assets part to learn how to add pictures
You have to note that those icons require being white in order to be of the desired color. But of course you can chose to change the established order.

- Color: Depends on the sector the skill is in.
Right= Red="231" Green="103" Blue="0" Alpha="255"
Center= Red="148" Green="168" Blue="0" Alpha="255"
Left= Red="135" Green="178" Blue="170" Alpha="255"

/!\ IMPORTANT/!
If you choose to put a skill in a sector which already has a skill in it you might need to move a bit this already existing skill. Best thing to do is to recover the name of this skill. Look at the localization file and search the in-game name. For example: Strength of the Wild will be %HeroSkillLeaderBattle10InfantryTitle. It means that the UnitSkillGuiElement

Name="HeroSkillLeaderBattle10Infantry". You will have to add it in your GuiElements xml in order to override what is in the game and change the Angle according to your need.

Create the skill reference

Ref: Simulation\UnitSkills[Hero].xml

- Name: The skill's "id", serves as a tag to identify it in any xml file. Has to be the same as the Gui element.

- Category and SubCategory: Category is always "UnitSkill" but there are different subcategories of skills according to their purpose. It depends on the major bonus.

- SimulationDescriptorReference: You can choose the number of levels here, each making reference to the effects descriptor (see below where to create effects).

- Prerequisite: If you want to link the skill to another it is the way to do so. You can also hide or allowed some skill for very specific hero (based on faction or genre for instance).

Create the effects

Ref: Simulation\SimulationDescriptors[SkillsHero].xml

- Name: The skill's "id", serves as a tag to identify it in any xml file. Has to be the name refered into the SimulationDescriptorReference of the UnitSkills[Hero].xml.

- Type: It is a "SkillHero" in this case.

- SimulationModifierDescriptors: Those will be the formulas used to create the effects. It is written as follows.

```
  <SimulationModifierDescriptors>
   <SimulationModifierDescriptor TargetProperty="AttributeAttack"  Operation="Addition"
Value="30"   Path="UnitHero" />
   <SimulationModifierDescriptor TargetProperty="AttributeDefense  Operation="Addition"
Value="30"   Path="UnitHero" />
  </SimulationModifierDescriptors>
```

TargetProperty: It is the origin of the applied effect.
Operation: It can be a multiplication, addition, percentage, subtraction, division or power.
Value: The number taken into account for the operation.
Path: The target of the effect.

Add the new object for the AI

Ref: AI\AIParameters[UnitSkill].xml

For each hero skill, you must write an AIParameterDatatableElement (as explained in the AI part).

Each AIParameterDatatableElement must contain these 6 AIParameters:

- AICity
- AIArmy
- AIExploration
- AIYoungCityAssignment
- AIArmyAssignment
- AIMatureCityAssignment

The three first AIParameters define the category of the skill. The three last AIParameters define where the AI should assign a hero with this skill.

How to change a resource

Due to some limitation with the world generator you will be able to change a resource only. No possibility to add one. Do not forget you won't be able to change to 3D model either.

The Gui element

For the resource itself
Ref: Gui\GuiElements[GameVariables].xml

- Name: The resource name, serves as a tag to identify it in any xml file. Each resource have different names (eg: Luxury1, Luxury2, Strategic3, etc.). There are 15 Luxury resources and 6 Strategic resources.

- Title: If you want your resource to be localized, use"%" to reference a localization key (eg: %ResourceTitle) and go to a). Otherwise skip to b).

a) Localized text version: Open EF_Localization_Assets_Locales.xml in MyMod\Localization\english (or other languages). Create a new LocalizationPair with the %name (eg: %ResourceTitle) and the actual name (eg: The new Resource). Translate appropriately in the other Localization_Locales files.

b) Simply write the Resource's name

- Description: Same as Title.

- Icon: Please refer to the modding/assets part to learn how to add pictures
(be careful with the name of the resource small icon as it must be different from what is already existing)

- SymbolString and Color: This element is related to the small symbol you can see near a luxury or strategic resource. Unfortunately it cannot be changed at the moment because the info is in the game assets part that you cannot access.

For the deposit
Ref: Gui\GuiElements[PointOfInterestTemplates].xml
- Name: The resource deposit name, serves as a tag to identify it in any xml file. Each resource deposit have different names (eg: Luxury1, Luxury2, Strategic3, etc.). There are 15 Luxury resources and 6 Strategic resources.

- Title: If you want your deposit resource to be localized, use"%" to reference a localization key (eg: %DepositResourceTitle) and go to a). Otherwise skip to b).

a) Localized text version: Open EF_Localization_Assets_Locales.xml in MyMod\Localization\english (or other languages). Create a new LocalizationPair with the %name (eg: %ResourceTitle) and the actual name (eg: The new Deposit Resource). Translate appropriately in the other Localization_Locales files.

b) Simply write the Deposit Resource's name

- Description: Same as Title.

- Icon: Please refer to the modding/assets part to learn how to add pictures

For the extractor
Ref: Gui\GuiElements[PointOfInterestImprovement].xml

- Name: The extractor name, serves as a tag to identify it in any xml file. Each extractor have different names (eg: ExtractorLuxury1, ExtractorLuxury2, ExtractorStrategic3, etc.). There are 15 Luxury resources and 6 Strategic resources.

- Title: If you want your resource to be localized, use"%" to reference a localization key (eg: %ExtractorTitle) and go to a). Otherwise skip to b).

a) Localized text version: Open EF_Localization_Assets_Locales.xml in MyMod\Localization\english (or other languages). Create a new LocalizationPair with the %name (eg: %ExtractorTitle) and the actual name (eg: The new Extractor). Translate appropriately in the other Localization_Locales files.

b) Simply write the Extractor's name

- Description: Same as Title.

- Icon: Please refer to the modding/assets part to learn how to add pictures

The booster linked to the resource

Ref: Simulation\DepartmentOfPlanificationAndDevelopment+Constructibles[Booster].xml

- Name: The booster definition's "id", serves as a tag to identify it in any xml file. Has to be the name refered into the SimulationDescriptorReference of the technology definition.

- Duration: You can manage how many turn the booster will work.

- SimulationDescriptorReference: You have to enter the name of the effect you create. See the part "Create the effects" below for details.

- CustomCost: Will be the price to get the booster activated.

The effects of the booster

Ref: Simulation\SimulationDescriptors[Booster].xml

- Name: The booster effects' "id", serves as a tag to identify it in any xml file. Has to be the name refered into the SimulationDescriptorReference of the Booster definition.

- SimulationModifierDescriptor: Those will be the formulas used to create the effects. It is written as follows.

```
<SimulationModifierDescriptors>
  <SimulationModifierDescriptor TargetProperty="DistrictFood"  Operation="Addition"
Value="2"  Path="./ClassCity/TerrainTagFood"/>
  <SimulationModifierDescriptor TargetProperty="CityGrowth"  Operation="Percent"
Value="0.2"  Path="./ClassEmpire/ClassCity"/>
  <SimulationModifierDescriptor TargetProperty="CityApproval"  Operation="Addition"
Value="15"  Path="./ClassCity"/>
</SimulationModifierDescriptors>
```

TargetProperty: It is the origin of the applied effect.
Operation: It can be a multiplication, addition, percentage, subtraction, division or power.
Value: The number taken into account for the operation.
Path: The target of the effect.

How to add a quest

Please note that this modding part will rely on an example to explain the main structure of a quest.
If you want to have a global vision of what are the different elements at disposal you will have to go on the wiki for further details. [http://LINK]

Create the Gui element

Ref: Gui\GuiElements[Quests#NewQuest].xml.
Be careful, the name after # must be the same that the one in Endless Legend\Community\NewQuest\Quests.

- Name: The quest's "id", serves as a tag to identify it in any xml file. Each Quest should have different names (eg: Quest1, Quest2, etc.).

- Title: If you want your quest to be localized, use"%" to reference a localization key (eg: %QuestTitle) and go to a). Otherwise skip to b).

a) Localized text version: Open EF_Localization_Assets_Locales.xml in MyMod\Localization\english (or other languages). Create a new LocalizationPair with the %name (eg: %NewQuestTitle) and the actual name (eg: The new Quest). Translate appropriately in the other Localization_Locales files.

b) Simply write the Quest's name

- Description: Same as Title.

- Icon: Please refer to the modding/assets part to learn how to add pictures [4]

- Steps: Will be the way to show the step into the game journal.

The following parts will occur in one same document.
Ref: Quests\QuestDefinitions[NewQuest].xml

The quest definition

- Category and Subcategory: It will depend on the type of the quest.

- Cooldown: Number of turns before the quest might be triggered again.

- SkipLockedQuestTarget: In case the quest target a ruin it can lock the location for it own purpose only ("false") or not ("true").

The other numbers are the occurence during a game.

The tags

Will be the basic way the quest is triggered. Before even ensuring the quest is allowed to happen (thanks to a prerequisite check).
Several types of tags exist at this time. The most common are:
<Tags>Interact</Tags> trigger a quest as soon as you search a ruin
<Tags>Talk</Tags> trigger a quest as soon as you parley with a
<Tags>BeginTurn</Tags> trigger a quest at the beginning of a turn (this one is mandatory if you want to trigger a quest based on a current turn or after a specific change in game simulation)
<Tags>Chapter #</Tags> mean to make link between chapters
<Tags>#FactionType</Tags> trigger only according a specific faction
<TagsMainQuest</Tags> to mark a quest as a main one

The prerequisites

It is the way to define what is limiting or mandatory for the quest to trigger. It mainly concerns the state of the game or the type of faction you play.

```
<Prerequisites Target="$(Empire)">
 <PathPrerequisite
Flags="Prerequisite">EmpireTypeMajor/ClassResearch,TechnologyEra2</PathPrerequisite>
 <InterpreterPrerequisite Flags="Prerequisite">$Property(ClassEmpire:WarCount) ge
0</InterpreterPrerequisite>
 <PathPrerequisite Inverted="true"
Flags="Prerequisite">../EmpireTypeMajor,Necrophages</PathPrerequisite> <–It means that if
you play Necrophages quest cannot be triggered–>
</Prerequisites>
```

There is more info on prerequisite in another part of the document. Please refer to the summary table at the beginning.

The variables

It will regroup all the elements you want to interact with. It can be a village, a ruin, a resource, etc. It will follow the conditions you decide. But the logic is very thorough. For more info on the variable request logic you will have to go on the wiki for further details. [http://LINK]

You will sometime need to get the variables of all your cities in order to get the region of your city in order to get the point of interests' type "ruins" in it in order to pick one and choose it as a

location to go on your quest. This means that if you still don't have a city, the variables will work as a prerequisite and will prevent your quest from launching.

Here is an example where I need cities. But more generally, if I want to check the surrounding regions around my empire being on the same continent I will have to write down this:

```
<Var VarName="$EmpireCity">
  <Any>
    <From Source="$Empire.$Cities"/>
  </Any>
 </Var>
 <Var VarName="$EmpireCityRegion">
  <From Source="$EmpireCity.$Region"/>
 </Var>
 <Var VarName="$SurroundingRegions">
  <From Source="$EmpireCityRegion.$RegionWithNeighbourRegions">
    <Where>
      <FilterRegionIsOnSameContinent RegionContextVarName="$EmpireCityRegion"/>
    </Where>
  </From>
 </Var>
```

Now that I have retrieved the info I can have a look at the point of interest in the variable of the region that has been chosen by the system. I will get a variable of a ruin (QuestPOI) in order in one of my step to ask the player to search the ruin in question.

```
<Var VarName="$PointOfInterestToInspect">
  <Limit LimitMin="5" LimitMax="6">
    <From Source="$SurroundingRegions.$PointsOfInterest">
      <Where>
        <PathPrerequisite>PointOfInterestTypeQuestLocation</PathPrerequisite>
      </Where>
    </From>
  </Limit>
 </Var>
 <Var VarName="$PointOfInterestToInspectLocation">
  <From Source="$PointOfInterestToInspect.$Position"/>
 </Var>
```

And then if I want to give info on the POI in the text of the journal I will have to set a localization variable.

```
<LocalizationVar LocalizationKey="$QuestPOI" Source="$PointOfInterestToInspect"/>
```

The sequence

It is the main structure of the quest. It is mandatory that the entire sequence remains between the marks <Controller_Sequence> </Controller_Sequence>
There can be different step(s) (at least one) that you will be able to see in the game journal (thanks to GuiElements, details at the bottom). All the variables from before will be used here.

First I need to update the game journal to indicate this is the current step.

```
<Action_UpdateStep StepName="SideQuestPOI#New-Step1" State="InProgress"/>
```

Then if there is a ruin that will be targeted best is to lock it to avoid conflict with other quests that could retrieve this same variable.

```
<Action_LockQuestTarget TargetVarName="$PointOfInterestToInspect"
LockOption="Lock"/>
```

Let's go to the step itself.

```
<Action_AddQuestMarker TargetEntityVarName="$PointOfInterestToInspect" Tags="Ruins"
RevealUnexploredLand="true" MarkerVisibleInFogOfWar="true"
Output_QuestMarkerVarName="$QuestMarkerPointOfInterestToInspect"/>
<–The quest will require searching a ruin. Here is the way to add the marker (ray of dust) on it.–>

<Decorator_Inspect TargetEntityVarName="$PointOfInterestToInspect"
Output_InstigatorSimObjectVarName="$InspectionInstigatorSimObject"
PrerequisiteNotVerifiedMessage="%QuestPrerequisiteNotVerified" Initiator="Empire">
<–The Output_InstigatorSimObjectVarName is a variable created by this decorator and it will
allow you to check what is in the army which is searching the ruin–>

<ConditionCheck_Prerequisite>
<Prerequisites Target="$InspectionInstigatorSimObject">
<PathPrerequisite
Flags="Prerequisite">ClassArmy/ClassUnit,UnitHero</PathPrerequisite>
</Prerequisites>
</ConditionCheck_Prerequisite>
</Decorator_Inspect>
<–This is the check, looking if there is a hero unit in the searching army–>

<Action_RemoveQuestMarker
TargetQuestMarkerVarName="$QuestMarkerPointOfInterestToInspect" />
<Action_LockQuestTarget TargetVarName="$PointOfInterestToInspect"
LockOption="Unlock"/>
<Action_UpdateStep StepName="SideQuestPOI#New-Step1" State="Completed"/>
<Action_UpdateQuest State="Completed"/>
<–As soon as the decorator is validated, all the actions are instant. Do not forget to update the
step for each step and the quest when it is finished–>
```

*A word on Decorator_*

A decorator is some kind of a check that you did a specific action during the quest sequence. Here is a list of possible Decorator below:

- Decorator_BeginTurn (certainly the most used)
- Decorator_CaptureCity
- Decorator_ConstructionEnded
- Decorator_KillArmy

- Decorator_Inspect
- Decorator_Colonize
- Decorator_Talk
- Decorator_VillagesDestroyed
- Decorator_DistrictLevelUp
- Decorator_UnitSkillUnlocked
- Decorator_FactionAssimilated
- Decorator_VillagesPacified
- Decorator_BoosterActivated
- Decorator_QuestCompleted

For instance, using the last decorator:

```
<Controller_Loop LoopCount="25">
 <Decorator_QuestCompleted Initiator="Empire"
LinkedStepProgression="MedalEra5HeroesAlt-Step1"/>
</Controller_Loop>
```

The loop is checking that the decorator has been completed X times.
The decorator is checking that the Empire has completed a quest.

Of course, lots of decorators have different variables attached to them and sometimes some prerequisites. It really depends a lot on what you intend to do during the quest.
Here are 2 examples to present the concept:

- I want the player to a precise skill at a precise level on a precise hero to validate the step/quest:

```
<Decorator_UnitSkillUnlocked SkillNameVarName="$SkillToCheck"
SkillLevelVarName="$SkillLevelToCheck" TargetUnitVarName="$Architect"/>
(each variable has been retrieve in the “variables” part of the quest)
```

- I want the player to have a city hall level to validate the step/quest:

```
<Decorator_BeginTurn (optional attribute : AllowOnQuestStart="true") Initiator="Empire">
 <ConditionCheck_Prerequisite>
  <Prerequisites Target="Empire">
   <InterpreterPrerequisite
Flags="Prerequisite">$Property(EmpireTypeMajor/ClassCity/DistrictTypeCenter:Level) ge
1</InterpreterPrerequisite>
  </Prerequisites>
 </ConditionCheck_Prerequisite>
</Decorator_BeginTurn>
```

The reward

There is a droplist of elements specific to solo quests in **Droplists[SoloQuests].xml** where you can take rewards from. Or add new ones. For the moment the most common available are:
- DroplistAdvancedTechnologies
  > You will get a technology from the next available era
- DroplistTechnologies
  > You will get a technology not already unlocked from the current era

- DroplistDust
    > You will get an amount of Dust proportional to your technological advancement
- DroplistPrestige
    > You will get an amount of Influence proportional to your technological advancement
- DroplistItems
    > You will get a random item
- DroplistResources
    > You will get an amount of resources (luxury or strategic) proportional to your technological advancement

Of course you can create your own droplist on the same model but changing the values or the technologies as will.

Then you need to link the reward to a step and choose the number of reward itself.
Here is an example with the first step of a quest.

```
<Steps>
 <Step Name=" SideQuestPOI#New- Step1">
  <Reward Droplist=" DroplistAdvancedTechnologies " Picks="1"/>
 </Step>
</Steps>
```

You can also write a text if you have given the reward through an action during the step sequence.
Pacification is done this way.

```
<Reward LocalizationKey="%Pacification"/>
```

How to add a victory

The Gui element

For the resource itself
Ref: Gui\GuiElements[VictoryConditions].xml

- Name: The name of the victory. Each victory should have a different name (eg: Victory1, Victory2, Victory3, etc.).

- Title: If you want your resource to be localized, use"%" to reference a localization key (eg: %VictoryTitle) and go to a). Otherwise skip to b).

a) Localized text version: Open EF_Localization_Assets_Locales.xml in MyMod\Localization\english (or other languages). Create a new LocalizationPair with the %name (eg: % VictoryTitle) and the actual name (eg: The new Victory). Translate appropriately in the other Localization_Locales files.

b) Simply write the Victory's name

- Description: Same as Title.

Note: No need to add picture information here. It will be automatically created according to the victory condition name.
Example: Name="Hero"
Image name in Resources\GUI\DynamicBitmaps\Factions folder of your mod will be: majorFaction(FactionName)MoodVictoryHero
(FactionName) should be replaced by each faction name in the game (because by default each will be able to trigger the victory).

The victory condition

Ref: Statistics and Achievements\VictoryConditions.xml

It is the place to define what is mandatory for the victory to trigger. It mainly concerns the expression of the victory itself and the alert(s) before the trigger happens.

```
<InterpreterBasedVictoryCondition Name="Hero" Category="Victory" SubCategory="Hero">

<Expression>($Property(./ClassEmpire/ClassUnit,UnitHero:LevelDisplayed) ge 30) or
($Property(./ClassEmpire/Garrison/ClassUnit,UnitHero:LevelDisplayed) ge 30)</Expression>

<Alert Repeat="false">($Property(./ClassEmpire/ClassUnit,UnitHero:LevelDisplayed) ge 20)
or ($Property(./ClassEmpire/Garrison/ClassUnit,UnitHero:LevelDisplayed) ge 20)</Alert>
```

Here I ask to have at least 1 hero in your empire at level 30, with an alert when one reach level 20.

For the alert the messae loc key is generated automatically according to the following pattern:
Message for you
%NotificationVictoryConditionAlertHero#0TitleYou
%NotificationVictoryConditionAlertHero#0DescriptionYou
Message for the others
%NotificationVictoryConditionAlertHero#0Title
%NotificationVictoryConditionAlertHero#0Description

Just change “Hero” to the name of your own victory (from xml reference) and the number #0 corresponds to the alert 0, then 1, then 2, according to the number of Alert you created.

The second part visible in this document is a bit less important.

```
<Progression Format="%VictoryProgressHero" SortVariable="Value">
 <Var Name="Value">$(GlobalScore)</Var>
 <Var Name="TargetValue">$(HighestGlobalScore)</Var>
 <Var Name="RemainingTurns">$Property(LastTurn) - $(Turn)</Var>
</Progression>
```

It is related to the statistics you will see on the score screen after victory completion.

Activation of a victory

Ref: Options\ GameOptionDefinitions.xml

Each victory can be activated/deactivated in the advanced option menu.
You must add your victory to this list too.

```
<GameOptionDefinition Name=”Hero” Default=”True” Category=”Game”
SubCategory=”VictoryConditions” GuiControlType=”Toggle”>
  <Tags>Advanced</Tags>
  <ItemDefinition Name=”False”/>
  <ItemDefinition Name=”True”/>
</GameOptionDefinition>
```

You just need to make a link with the name of your own victory and it should be enough to do the trick.

How to add or change text

Adding new text

You've created a mod, but you want to add your own texts?

1) Whenever you see text (like titles and descriptions) in your modified xml, instead of directly writing the text in that space, use an id like %YourText.

2) Open or create Documents\Endless Legend\Community\YourMod\Localization\AnyLanguage\EF_Localization_Assets_Locales.xml. In the idea that you only add new text you should erase all the content in order to only keep the following lines:

```
<?xml version="1.0" encoding="utf-8"?>
<Datatable xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

</Datatable>
```

3) Add a new LocalizationPair entry with %YourText between the Datatable marks.

```
<?xml version="1.0" encoding="utf-8"?>
<Datatable xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <LocalizationPair Name="%SomethingTitle"></LocalizationPair>
  <LocalizationPair Name="%SomethingDescription"></LocalizationPair>
</Datatable>
```

4) Enter your text between the tags.

```
<?xml version="1.0" encoding="utf-8"?>
<Datatable xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <LocalizationPair Name="%SomethingTitle">Something Powerful</LocalizationPair>
  <LocalizationPair Name="%SomethingDescription">Something so powerful we do not even
what it is for now.</LocalizationPair>
</Datatable>
```

5) Report the change in the other languages.

Changing an already existing text

You've created a mod, but you want to change some texts, descriptions, etc?

1) Whenever you see text (like titles and descriptions) you want to change, you must get their reference name in-game through Steam\SteamApps\common\Endless Legend\Public\Localization\ AnyLanguage

2) Open or create Documents\Endless Legend\Community\YourMod\Localization\AnyLanguage\EF_Localization_Assets_Locales.xml or EF_Localization_Locales.xml. It depends in what in-game files you have seen the text you want to change. For instance:

```
<LocalizationPair Name="%UnitBrokenLordsBlackBishopTitle">Dust Bishop</LocalizationPair>
```

It can become:

```
<LocalizationPair Name="%UnitBrokenLordsBlackBishopTitle">Dust Archbishop</LocalizationPair>
```

4) Report the change in the other languages.

Forcing a tooltip effect to show your text

You've created a mod with a city improvement or a technology and you want to write down the effect by yourself?

Create a GuiElement:
1) Ref for technology: **GuiElements[TechnologyUnlock].xml**
Ref for city improvement: **GuiElements[PointOfInterestImprovement].xml**

Add an ExtendedGuiElement. Be careful the name must be exactly the same as the simulation descriptor of the city improvement or technology you want to override.

```
<ExtendedGuiElement Name="NewElementTooltip">
  <TooltipElement Ignore="true">
    <EffectOverride>%NewElementTooltipEffect</EffectOverride>
  </TooltipElement>
</ExtendedGuiElement>
```

For instance, if I want to override the technology Signal Corps I will write the Name="TechnologyArmySize2" according to the localization key it is bind to.

2) Override the tooltip with your own localization key as you can see above.

How to add or change 2D assets

Please know that pictures are mainly in png format and you should stick to this.

Create a new palette of faction color

Ref: Mapping\Palettes.xml

- The name must be “Standard” in order to override the main palette of the game.

- ColorReference: You have to add a color reference but you can also use the name of an existing one to replace it.

Then you just have to find the RGB reference of your color. Add them on the line.
<Color Red="56" Green="73" Blue="204" Alpha=" 255"/>
The Alpha part should remain at 255.

NOTE: There is a limit of 10 colors for the mod to correctly work.

About changing already existing pictures

The following information is very important to catch in order to change properly a picture that is already present in-game.
You will found out that if you try to change a large picture and a small picture just by following the structure of the game and keeping the in-game name (example with units):
Documents\Endless Legend\Community\”ModName”\Resources\GUI\DynamicBitmaps\Units\UnitMadFairiesArcherLarge
Documents\Endless Legend\Community\”ModName”\Resources\GUI\AtlasedBitmaps\Units\UnitMadFairiesArcherSmall
Then the large picture will be yours but the small one will remain the same old.

- Why? All the AtlasedBitmaps are generated at the beginning of the game on a big texture and will never be computed again.
- What to do? You need to change the name of the picture and the GuiElement accordingly.
- Best to do? We recommend you to create in the resources folder a simple “pictures” folder to put everything in it: Community\”ModName”\Resources\Pictures\Units

```
<Icon Size="Small" Path="Pictures/Units/UnitMFSettlerSmall" />
<Icon Size="Large" Path="Pictures/Units/UnitMadFairiesSettlerLarge" />
```

That is true for a lot of the small picture in the game. So if you stumble on the problem just think about changing the picture name in correct the GuiElement file.

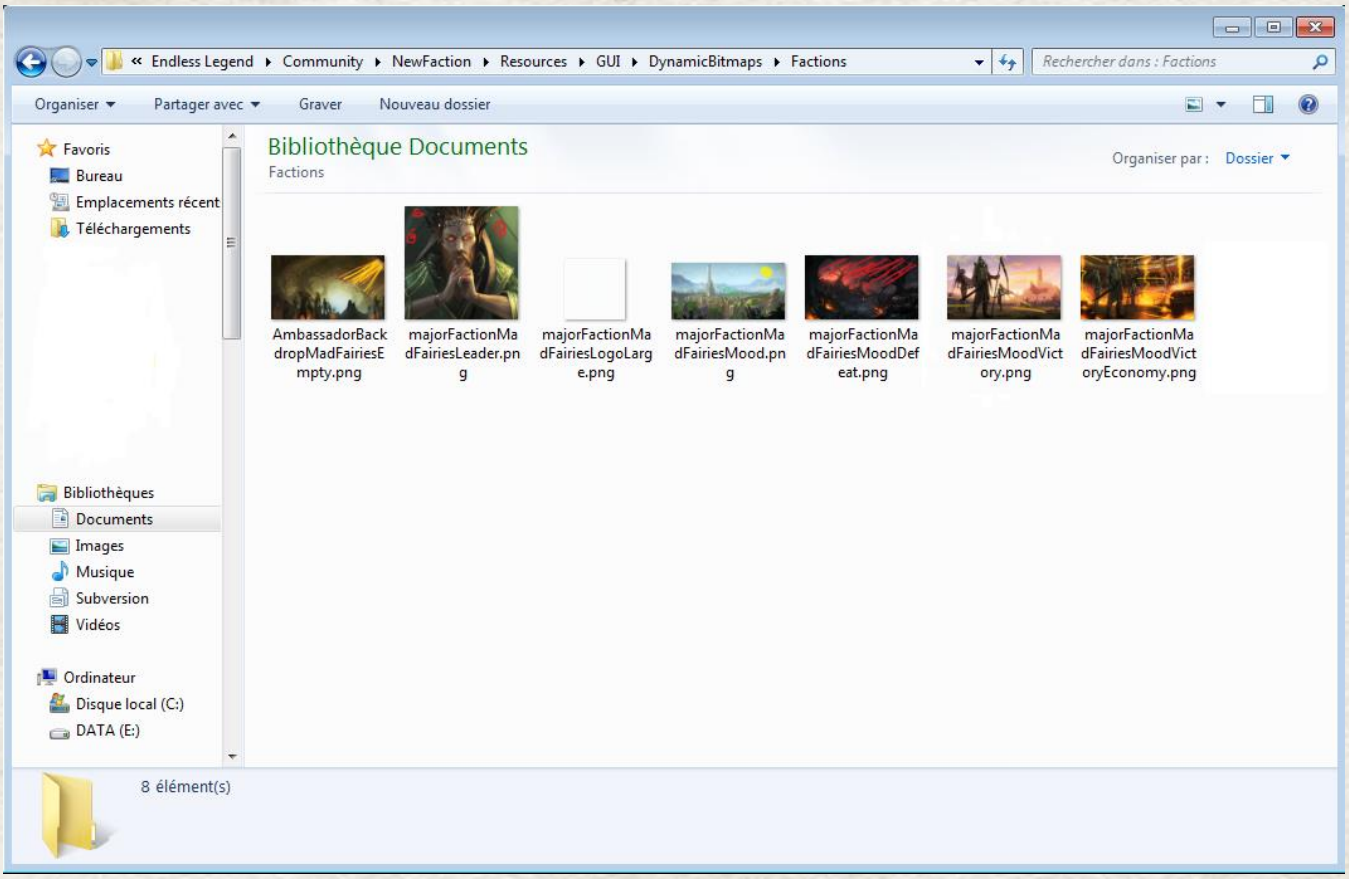
Adding new pictures for a faction

In fact you just have to add some pictures into a folder following the game structure and the name of the pictures.

Here are the corresponding names of the factions:
Wild Walkers = MadFairies / Broken Lords = BrokenLords / Vaulters = Vaulters / Mezari = Mezari / Necrophages = Necrophages / Ardent Mages = RageWizards / Roving Clans = RovingClans / Drakken = Drakkens / Cultists = Cultists

To change the images of a faction the path should be the same as in-game (to avoid changing GuiElement): Documents\Endless Legend\Community\"ModName"\Resources\GUI\DynamicBitmaps\Factions

Here is the list of pictures taking WildWalkers (MadFairies) as example:
- majorFactionMadFairiesLeader.png: Picture of the leader
- AmbassadorBackdropMadFairiesEmpty.png: Background image of your ambassador
- majorFactionMadFairiesLogoLarge.png: Logo of your faction (visible for your armies mainly)
- majorFactionMadFairiesMood.png: Small picture above faction traits in New Game screen
- majorFactionMadFairiesMoodDefeat.png: Picture shown when you are defeated
- majorFactionMadFairiesMoodVictory.png: Main picture for victory



> You can add the following suffix in order to change the image of a specific victory: Economy / Expansion / GlobalScoreWhenLastTurnReached / LastEmpireStanding / Supremacy / Wonder / MostTechnologiesDiscovered

If you want **to change the picture of units** it is pretty much the same. What you have to know is that there are 2 different formats, large icons (256x128 px) and small icons (96x48 px).
You will need to change the path and name (at least for the small icon) in not less than 3 GuiElement files:
Ref: GuiElements[BodyDefinitions].xml, GuiElements[UnitDesigns#Mercenaries].xml, GuiElements[UnitDesigns].xml

Adding new pictures for a city improvement

When you add or change a city improvement you should have a **GuiElements[CityImprovement].xml** file in reference (in your Gui modding folder) with the following look.

```
<GuiElement Name="CityImprovementNew">
  <Title>%CityImprovementNewTitle</Title>
  <Description>%CityImprovementNewDescription</Description>
  <Icons>
    <Icon Size="Small" Path=" Pictures/Improvements/CityImprovementNewSmall" />
    <Icon Size="Large" Path=" Pictures/Improvements/CityImprovementNewLarge" />
  </Icons>
</GuiElement>
```

In order for the icons to be in the game after your mod is installed and launched you need to create a folder which includes the Large icons (256x128 px) and the Small icons (96x48 px).

Adding new pictures for a technology

When you add or change a technology you should have a **GuiElements[Technology].xml** file in reference (in your Gui modding folder) with the following look.

```
<TechnologyGuiElement Name="TechnologyDefinitionNew" X="-95" Y="215">
  <Title>%TechnologyDefinitionNewTitle</Title>
  <Description>%TechnologyDefinitionNewDescription</Description>
  <Icons DefaultSize="">
    <Icon Size="Small" Path="Pictures/Technology/TechnologyDefinitionNewSmall" />
    <Icon Size="Large" Path="Pictures/Technology/TechnologyDefinitionNewLarge" />
  </Icons>
  <TechnologyEraReference Name="TechnologyEraDefinition1" />
</TechnologyGuiElement>
```

In order for the icons to be in the game after your mod is installed and launched you need to create 1 folder which includes the Large icons (256x128 px) and Small icons (96x48 px).

Adding new pictures for an item

When you add or change an item you should have a **GuiElements[ItemDefinitions].xml** file in reference (in your Gui modding folder) with the following look.

```
<GuiElement Name="ItemHeadIronNew">
   <Title>%ItemHeadIronNewTitle</Title>
   <Description>%ItemHeadIronNewDescription</Description>
   <Icons>
<Icon Size="Small" Path="Pictures/EquipmentItems/ItemHeadIronNewSmall" />
<Icon Size="Large" Path="Pictures/EquipmentItems/ItemHeadIronNewLarge" />
   </Icons>
 </GuiElement>    <Icons DefaultSize="">
     <Icon Size="Small" Path="Pictures/Technology/TechnologyDefinitionNewSmall" />
     <Icon Size="Large" Path="Pictures/Technology/TechnologyDefinitionNewLarge" />
   </Icons>
   <TechnologyEraReference Name="TechnologyEraDefinition1" />
 </TechnologyGuiElement>
```

In order for the icons to be in the game after your mod is installed and launched you need to create 1 folder according the path below:
Documents\Endless Legend\Community\NewTechnology\Resources\Pictures\Technology

It includes the Large icons (256x128 px) and the Small icons (96x48 px).

Adding new pictures for a skill

When you add or change a skill you should have a **GuiElements[UnitSkill].xml** file in reference (in your Gui modding folder) with the following look.

```
 <UnitSkillGuiElement Name="HeroSkillLeaderBattle19" Sector="0" Angle="0" Radius="1">
   <Title>%HeroSkillLeaderBattle19Title</Title>
   <Description>%HeroSkillLeaderBattle19Description</Description>
   <Icons>
     <Icon Size="Small" Path=" Pictures/HeroSkills/HeroSkillLeaderBattle19Small" />
     <Icon Size="Large" Path=" Pictures/HeroSkills/HeroSkillLeaderBattle19Large"/>
   </Icons>
   <Color Red="135" Green="178" Blue="170" Alpha="255" />
 </UnitSkillGuiElement>
```

In order for the icons to be in the game after your mod is installed and launched you need to create 1 folder which includes the Large icons (256x128 px) and the Small icons (48x48 px).
In order for the icon to fit in the game it should be a white image with transparency around it (a color is applied through xml files).

Please note that if you want it can work with only 1 folder and giving it other name. We just show the right game structure as a reference here.

Adding new pictures for a resource

When you want to change a resource you should have a **GuiElements[GameVariables].xml** file in reference (in your Gui modding folder) with the following look.

```
<ExtendedGuiElement Name="Luxury5">
  <Title>%LuxuryNewTitle</Title>
  <Description>%LuxuryNewDescription</Description>
  <Icons>
    <Icon Size="Small" Path="Pictures/GameVariables/LuxuryNewHD" /> <–It is mandatory that the name is different from an already existing resource. Otherwise it will not override the previous image.–>
    <Icon Size="Large" Path="Pictures/Resources/Luxury5Large"/>
  </Icons>
  <SymbolString>\7725\</SymbolString>
  <Color Red="255" Green="255" Blue="255" Alpha="255" />
</ExtendedGuiElement>
```

In order for the icons to be in the game after your mod is installed and launched you need to create 1 folder which includes the Large icons (256x128 px) and the Small icons (24x24 px).
In order for the icon to fit in the game it should be a white image with transparency around it (a white color is applied through xml files).

Please note that if you want it can work with only 1 folder and giving it other name. We just show the right game structure as a reference here.

To change the resource deposit or resource extractor it is the same process as adding a 2D asset for an item or city improvement.

The change on the Icon path are to be made in both files:
GuiElements[PointOfInterestImprovement].xml (extractor)
GuiElements[PointOfInterestTemplates].xml (deposit).

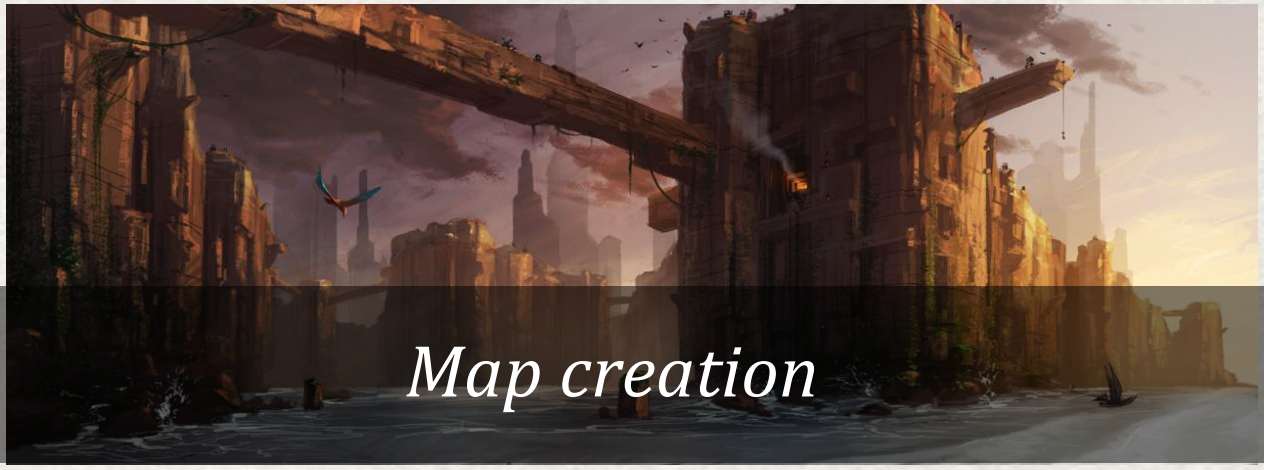
Changing pictures of a hero

Considering you cannot add new heroes for the moment due to limitation of 3D model addition you do not even need to change an xml file to override an existing portrait of a hero.

In order for the icons to be in the game after your mod is installed and launched you need to create 1 folder which includes the Large icons (256x128 px) and the Small icons (96x48 px).

You can have up to 6 heroes per faction (sometime a hero from a quest can exist and be a 7th one).
Name of the image should override existing ones. Follow the example:
BrokenLordsHero1Large.png / BrokenLordsHero1Small.png

Do not forget that names of the factions can change in xml from what is written in game:
Wild Walkers = MadFairies / Broken Lords = BrokenLords / Vaulters = Vaulters / Mezari = Mezari / Necrophages = Necrophages / Ardent Mages = RageWizards / Roving Clans = RovingClans / Drakken = Drakkens / Cultists = Cultist



Map creation

There is a possibility to create new maps that will be activated during a new game through the generation presets system.

A tutorial with files to download as examples is available on our forums:
http://forums.amplitude-studios.com/showthread.php?84218-Map-Creation-Tutorial

We encourage you to get all the information there.

Advanced: Simulation Descriptor

During the different examples in this modding tutorial you have been able to work on a lot of SimulationDescriptors xml.
Here are some additional details on this specific subject.
To note: A very usable xml is Simulation\SimulationDescriptors[Class].xml as you can see in it some of the main classes and properties associated.

Simulation descriptor structure

A simulation descriptor contains a list of property descriptors and a list of modifier descriptors.

```
<SimulationDescriptor Name="ClassUnit" Type="Class">
  <SimulationPropertyDescriptors>
    <!– <SimulationPropertyDescriptor/> –>
  </SimulationPropertyDescriptors>
  <SimulationModifierDescriptors>
    <!– <SimulationModifierDescriptor/> –>
    <!– <BinarySimulationModifierDescriptor/> –>
  </SimulationModifierDescriptors>
</SimulationDescriptor>
```

Add a property descriptor

Property descriptor parameters

Parameter	Default value	Description
Name	No default value	The name of the property. This parameter must be unique!
BaseValue	0	The base value of the property.
IsSealed	false	Define if the property is sealed or not. A sealed property can't have modifiers.
MinValue	0	Define the minimum value of the property. (set to 'Negative' to define MinValue to -Infinity).
MaxValue	Infinity	Define the maximum value of the property.

Composition None Define the composition method of the property.

Property composition methods

Method	Description
None	No composition.
Maximum	Get the maximum value of their children.
Minimum	Get the minimum value of their children.
Sum	Get the sum of the children values.

Examples:

```
<SimulationPropertyDescriptor Name="Dust" Composition="Sum"/>
<SimulationPropertyDescriptor Name="NetGrowth" MinValue="Negative"/>
```

Add a modifier descriptor

Modifier descriptor parameters

Parameter	Default value	Description
IsBindOnSource	true	Define where the property Value is computed. If true, the Value is computed from the source object. If false, the Value is computed from the target object defines by the path.
Operation	Addition	Define the arithmetic operation used by the modifier. See below for details.
Path	No default value	Define the target object of the modifier.
Priority	0	As the operation determine the order of the modifier and as sometimes you want to perform a multiplication before an addition, priorities allow forcing an order. The more the priority is small the more it will be done first... Within the priority, the operation order remains.
TargetProperty	No default value	Define the name of the property to modify.
ValueMustBePositive	false	Most of the time used with a Percent operation. The modifier will affect the property value only if the current property is positive.

Modifier operations

Operation
Addition
Subtraction
Multiplication
Division
Power
Percent

The operations are done in this order:
BaseValue of property
+ All addition/substraction
+ All Multiplication/Division
+ All power
= Intermediate value

Then we apply the percent modifiers based on intermediate value. This is to avoid appliying the percent on another percent:
If intermediate value = 100 and we have one modifier with +10% and another with +20% the result have to be: 100 * (1 + 0.1 + 0.2) and not 100 * 1.1 * 1.2

A “Force” operation will override all the other operations of the same Priority. If you use two force modifiers on the same property, only the first one will be active. If you use force and another operation in the same priority range it won’t work either…
Force will "force" a value to the property. And this, no matter of the other operation applied before or during the same priority.
So, for instance, if you want to reset a value to 0 and then modify it we would recommend something like that:

```
 <SimulationDescriptor Name="NameOfYourDescriptorHere">
  <SimulationModifierDescriptors>
   <SimulationModifierDescriptor Path="YourPathHere" TargetProperty="TheTargetedProperty"
Operation="Force" Value="0" Priority="5"/>
   <SimulationModifierDescriptor Path="YourPathHere" TargetProperty="TheTargetedProperty"
Operation="Subtraction" Value="0.1" Priority="6"/>
  </SimulationModifierDescriptors>
 </SimulationDescriptor>
```

So first you force the property to 0 and then you substract 0.1. I used the priority 5 to be sure to be after anything else.

Single modifier descriptor parameters

Parameter	Default value
Value	No default value

Examples:

```
<!– Money += NetDust –>
<SimulationModifierDescriptor TargetProperty="Money" Operation="Addition"
Value="$(NetDust)"/>
```

Single modifier descriptor parameters

Parameter	Default value
Left	No default value
BinaryOperation	No default value
Right	No default value

Examples:

```
<!– NetDust += RawDust - DustUpkeep –>
<BinarySimulationModifierDescriptor TargetProperty="NetDust" Operation="Addition"
Left="$(RawDust)" Right="$(DustUpkeep)" BinaryOperation="Subtraction"/>
```

```
<!– Offense += X% of Offense with X = Morale * 0.1 –>
<BinarySimulationModifierDescriptor TargetProperty="Offense" Operation="Percent"
Left="$(Morale)" BinaryOperation="Multiplication" Right="0.1" />
```

Count modifier descriptor parameters

Parameter	Default value
Value	No default value
CountPath	No default value

Examples:

```
<!– A city improvement give +4 CityScience to the City for each Anomaly in the city. –>
<CountSimulationModifierDescriptor TargetProperty="CityScience" Operation="Addition"
Value="4" Path="./ClassCity" CountPath="ClassCity/TerrainTagAnomaly"/>
```

Advanced: Interpreter Prerequisites

PathPrerequisite

During the different examples in this modding tutorial you have been able to see a lot of PathPrerequisite which allow retrieving straight information from the game simulation.

<PathPrerequisite Flags="Prerequisite,Discard">.../ClassEmpire/Uniques,CityImprovement8</PathPrerequisite>

With InterpreterPrerequisite you can look at checking values or specific situations in the game simulation.

InterpreterPrerequisite

Here are the functions you can use.

Function	Description
$**Count**(simulationPath)	Return the number of simulation objects that valid the path *simulationPath*.
$**Path**(simulationPath)	Return true if the *simulationPath* is valid.
$**Property**([simulationPath:]propertyName)	Return the value of *propertyName*.
$**Descriptor**([simulationPath:]descriptorType)	Return the name of the descriptor of type *descriptorType*.
$**HasDescriptor**([simulationPath:]descriptorType)	Return true if the context simulation object have a descriptor of type *descriptorType*.
$**Link**(variableName|functionName|functionParams)	Return the value returned by the interpreter function *functionName* (called with parameters *functionParams*) if the variable in context named *variableName* is a simulation object.
$**PropertyFilteredCount**([simulationPath:]propertyName;operator;value)	Return the number of simulation objects that valid the path *simulationPath* AND the property condition (*propertyName operator value*).
$**SumProperty**([simulationPath:]propertyName)	Return the sum of the value of *propertyName*.

$MaxProperty([simulationPath:]propertyName)	Return the maximum value of *propertyName*.
$MinProperty([simulationPath:]propertyName)	Return the minimum value of *propertyName*.

Examples:

<InterpreterPrerequisite Inverted="false">($Property(ClassEmpire:MaximumLuxury1Count) - $Property(ClassEmpire:NetLuxury1)) eq 0</InterpreterPrerequisite>

<AIParameter Name="PopulationRatio" Value="$Property(ClassCity:Population) / $MaxProperty(../ClassEmpire/ClassCity:Population)"/>

<InterpreterPrerequisite Inverted="false">$PropertyFilteredCount(EmpireTypeMajor/ClassCity/DistrictTypeDistrict:Level;ge;1) ge 2</InterpreterPrerequisite>

And here is the info about operators:

Operator	Precedence	Description
*	5	Multiply
/	5	Divide
^	5	Power
+	4	Add
-	4	Subtract
eq	3	Equals
ne	3	Not equals
not	3	Not
min	3	min operand value
max	3	max operand value
gt	2	Greater than
ge	2	Greater or equals
lt	2	Less than
le	2	Less or equals
or	1	Or
ornot	1	Or not
and	0	And
andnot	0	And not

Advanced: AI Parameter

Add a AIParameterDatatableElement

The AI cannot read the simulation files. For each element from the simulation, it needs an object to translate the gameplay data into the AI's language. This object is called "AIParameterDatatableElement".

An AIParameterDatatableElement contains parameters called "AIParameter". Each parameter translates a gameplay data into an AI data.

For instance, this is the AIPArameterDatatableElement corresponding to the Seed Storage technology:

```
<AIParameterDatatableElement Name="TechnologyDefinitionFood0">
   <AIParameter Name="CityFood"            Value="10"/>
   <AIParameter Name="CityFoodPercent"     Value="0.15"/>
</AIParameterDatatableElement>
```

You can create as many AIParameters as you want. However, new AIParameters must be converted into known AIParameters with another kind of object: the "AIParameterConverter". If you don't want to write new AIParameterConverters, you can use the existing AIParemeters. To understand how to write an AIParameterConverter, please see below.

Add a AIParameterConverter

Ref: AI\AIParameterConverters[Empire].xml

An AIParameterConverter allows the AI to convert an unkown AIParameter into another AIParameter. For instance, this AIParameterConverter converts BaseSciencePerPopulation (science per population) into two AIParameters:

```
<AIParameterConverter Name="BaseSciencePerPopulation">

   <ToAIParameter AIParameter="AICityResearch">$(Input) *
$Property(SciencePopulation)</ToAIParameter>
```

```
    <ToAIParameter AIParameter="AIEmpireResearch">$(Input) *
$SumProperty(ClassEmpire/ClassCity:SciencePopulation)</ToAIParameter>

</AIParameterConverter>
```

The value $(Input) is the initial value of BaseSciencePerPopulation.

Balance the global strategy

Ref: AI\AIStrategicPlanDefinition[Empire].xml

This file contains weights for the AIParameters used at a global scale (empire plans, research, colonization…). You can increase values to make the AI give more attention to the concerned AIParameters, or decrease the values to have the opposite effects. Depending on the progression of the game, the AI changes its current empire state to adapt to the global situation. Each empire state contains a full set of values for the AIParameters. Here are the different empire states:

- START: beginning of the game
- ENHANCE: the AI just built a new city → the AI tries to improve its cities, especially the young ones
- EXPAND: all AI's cities are mature or its population reaches a threshold → the AI tries to build new cities
- CAPTURE: the AI wants to declare war to another empire, or is already in war → the AI tries to attack another empire to capture its cities
- DEFENSE: the AI is in danger → the AI tries to reinforce its cities

Balance the local strategy

Ref: AI\AICityStates.xml

The file works exactly like the **AIStrategicPlanDefinition[Empire].xml** except it concerns cities management instead of the empire. To understand how works the empire management, please read above.

Here are the different city states:

- FIRST STEPS: the city is young → the AI focused its city on food and industry
- BALANCE: the city is mature → the AI tries to have a balanced city
- WAR BORDER: there is a war against an empire with common frontiers with the city →the AI tries to produce defenses and units with this city
- BESIEGED: the city is besieged → the AI tries to produce defenses and units with this city
- FORGE: the city produces more industry than others and is mature → the AI specializes the city into industry and units production

The second part of the file also contains similar states only used when the players chose to automate constructions in their cities.

Advanced: Global Quests

Until now, variables, tags and prerequisites were dedicated to your own empire.
Here are some elements that will change this rule and open the world of Auriga to global quests, competitive or cooperative ones.

Quest Definition

Most of the references in quest definition must become “global”.

Example with a cooperative quest:

```
IsGlobal="true" Category="GlobalQuest" SubCategory="Cooperative" Cooldown="0"
GlobalCooldown="15" NumberOfGlobalQuestConcurrentInstances="-1"
```

Otherwise, the only real difference with solo quests will be the way to complete it.
In order to indicate this you have to add to the QuestDefinition: GlobalWinner="Participants" or "First".
First = Competitive objective
Participants = All are rewarded according to their commitment.

Global variables and prerequisites

There is a new target you can use in the prerequisites. It is to check something on at least one of all the empires. For instance, if one empire has reached the era 2:

```
<Prerequisites Target="$(Empires)">
    <PathPrerequisite
Flags="Prerequisite">EmpireTypeMajor/ClassResearch,TechnologyEra2</PathPrerequisite>
</Prerequisites>
```

Some specific variables named “InterpretedVar” have been added to be checked retrospectively in prerequisites.
Here is an example with a double check on number of active empires + number of technology unlocked:

```
    <InterpretedVar VarName="NumberOfEmpires" Target="$(Empires)"
UsedInPrerequisites="true">
      <Expression>$Count(EmpireTypeMajor,!EmpireEliminated)</Expression>
    </InterpretedVar>
    <InterpretedVar VarName="NumberOfTechsUnlocked" Target="$(Empires)"
UsedInPrerequisites="true">
<Expression>$Property(EmpireTypeMajor,!EmpireEliminated/ClassResearch:UnlockedTechnolo
gyCount)</Expression>
    </InterpretedVar>
```

Then an interpreter prerequisite has to be created in order to check the values in variables.

```
<InterpreterPrerequisite Flags="Prerequisite">$(NumberOfTechsUnlocked) ge (9 *
$(NumberOfEmpires))</InterpreterPrerequisite>
```

Otherwise, the system is similar to the solo quests and variables logic remains the same (sequences, step and GUI elements).

Rewards

Do not forget the rewards too. There is a specific droplist for them: **Droplists[GlobalQuests].xml**
Pretty the same than solo rewards but for cooperative quests you can make a difference for the best contributor if your steps require a Decorator activation.
Then the reward part will look like that:

```
  <Steps>
    <Step Name="GlobalQuestCoop#XXX-Step1">
      <ProgressionRange StartValue="0" EndValue="XX"/>
      <!–Per Contribution–>
      <Reward Droplist="DroplistCoopQuestDust" Progressive="true"/>
      <Reward LocalizationKey="%GlobalQuestCoopPerContributionReward"
JustForShow="true"/>
      <!–Best Contributor–>
      <Reward DropVar="$BestContributorRewardAmount" MinimumRank="1"/>
      <Reward LocalizationKey="%GlobalQuestCoopBestContributorReward"
JustForShow="true"/>
    </Step>
  </Steps>
```

About global events

Ref: Quests\QuestDefinitions[GlobalEvent].xml

The global events are effects that are randomly happening during the game.
We used the global quests system to create them, especially the cooperative side (GlobalWinner="Participants") in order to impact each player. Of course, some variables/prerequisites are global too in order to check the entire environment in which the event will trigger itself.

Otherwise, in the quest sequence there will be only 1 sentence that will look like this:

```
<Controller_Sequence>
 <Action_ApplyWorldEffect WorldEffectName="QuestWorldEffect#XXX" Duration="XX" />
       <Decorator_EndTurn />
</Controller_Sequence>
```

Name of the world effect and number of turns it happens are all yours.
What is important is that the world effect is referenced here:
SimulationDescriptors[QuestWorldEffects].xml

Then it is up to you to change anything you want from the simulation using SimulationModifierDescriptor. As usual do not hesitate to look at the things that have been done already.

And don't forget the GUI Elements.
Ref: GuiElements[QuestWorldEffects].xml
Aside from the picture, you must keep in mind that there are 2 notifications used: one with the effects happening, the other with the effects disappearing.
Examples are in the game as usual.