

Civ 6 Unit Asset Tutorials

Level 1 - Hello World

By Leugi

So... making units is pretty much a basic and much necessary knowledge for almost all possible Civilization VI mods; whether Leaders, Civilizations and even some cool mechanics. Through Civ 6 Asset Editor one can create unit assets that are varied thanks to the fact one can mix and match different unit parts whether custom or vanilla, so it offers a nice option for unit creation. However there's an issue, the asset editor is sometimes buggy and sometimes not very intuitive, and that can lead to needless frustration on the modding process. But fear not! For it's here where this tutorials will help you!

I've decided to split the tutorial in different levels that will come little by little. This is to aid in an easier comprehension of the unit making process, and also because later levels do require Photoshop/Gimp and Blender skills (but honestly, you don't need to get to the later levels most of the time). Each level even has cheesy tutorial level names!:

1. **Level 1 - Hello World:** Introduces the process of unit asset previewing, tints, and creating artdefines. This is the basic level.
2. **Level 2 - Change your behaviour!:** Introduces creating custom assets, changing the animations, adding xlp's, icons and mounted units.
3. **Level 3 - Material Needs:** Introduces materials and basic texture editing.
4. **Level 4 - A Model to Follow:** Introduces basic model editing for custom units, geo files and stuff.

This tutorial assumes you already know how to code your unit, so if you don't, try to learn that first; it also assumes you already configured your ModBuddy correctly, so again, do so please... So, without further ado, let's begin.

What you need:

- Civ 6 and Civ6 SDK Tools.
- Patience. Required to run the Asset Editor.

Loading an Asset

Opening the Asset Editor

So... let's say you already have your unit created. In this case, the Database Name of our unit is "UNIT_A_TEST" for the sake of simplicity. In a usual case you need to know what name your unit will have on the database (UNIT_....) since we'll need that later on.

I suggest you open this [project file](#) as this contains UNIT_TEST, a unit literally created for testing purposes with 4 movement, 80 melee combat and 80 ranged combat, without a prereq and that costs 10 production.

So the first thing we need to do is opening the Asset Editor. While on ModBuddy you can find the Asset Editor under the Tools submenu (assuming you installed everything well). So get ready to grab a drink and some potato chips, as the AE can sometimes take a very long time to load.



There. The Asset Editor is open! The next thing we need is to open a base Asset.

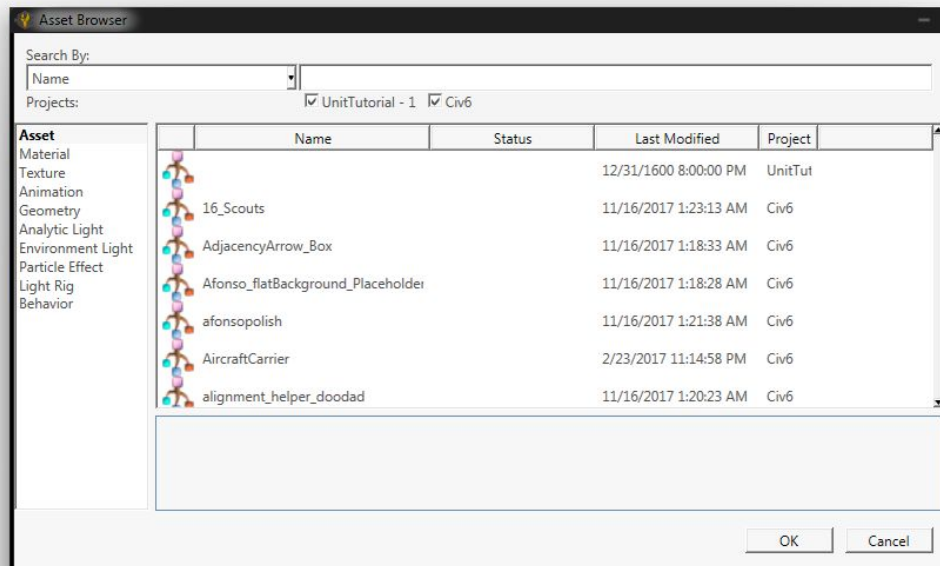
Base Asset Stuff

Assets are pretty much what units are made from. There are assets for body parts, armors, clothing, accessories, weapons, hats, horses, hair, etc. A unit is composed of these assets.

The most important of unit assets is the Armor asset. Armor pretty much refers to the base clothing of every unit (like the Knight's shining armor, or the Infantry generic uniform). They are the most important because **EVERY ASSET** is put on the armor (yes, even bodies), as

armor assets control the movement of the unit ingame. So the first asset we need to open is the base armor for our unit.

To load an asset go the File submenu -> **Open Entity**. The Asset Browser should appear:



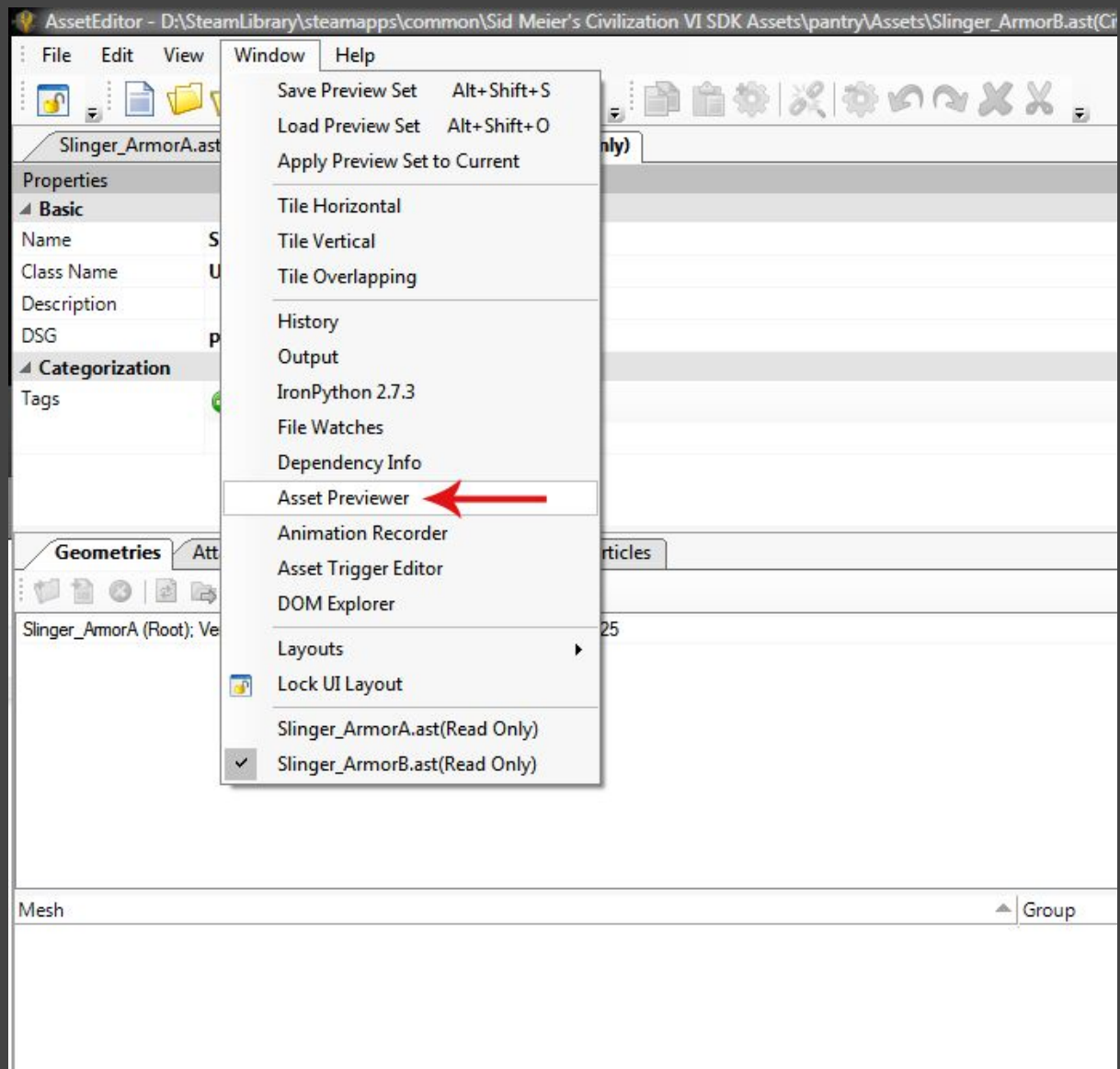
With the Asset Browser you can search for every single asset existing ingame (except DLCs because they hate us). When you make a custom unit you need to know which base armor to use, so checking many could be useful if you're undecided.

For the purposes of this tutorial search for Slinger_ArmorA and open it.

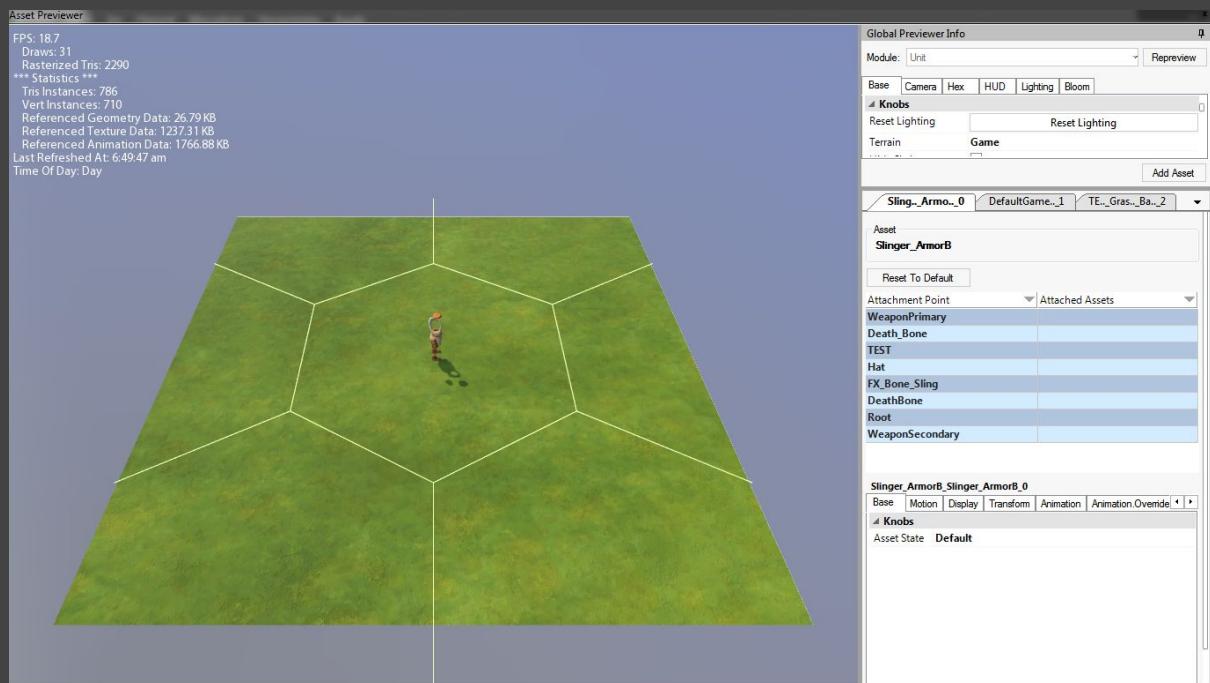
Preview the Asset

Introducing the Asset Previewer

the Asset Previewer. You can do this by clicking on the Window Submenu -> Asset Previewer. I suggest never leaving the Asset Previewer open for too long, since it can consume a lot of memory (as it is basically running the game engine on a little window as you mod)



Once you load the Asset Previewer (depending on the PC it could take a bit) you'll see something like this:



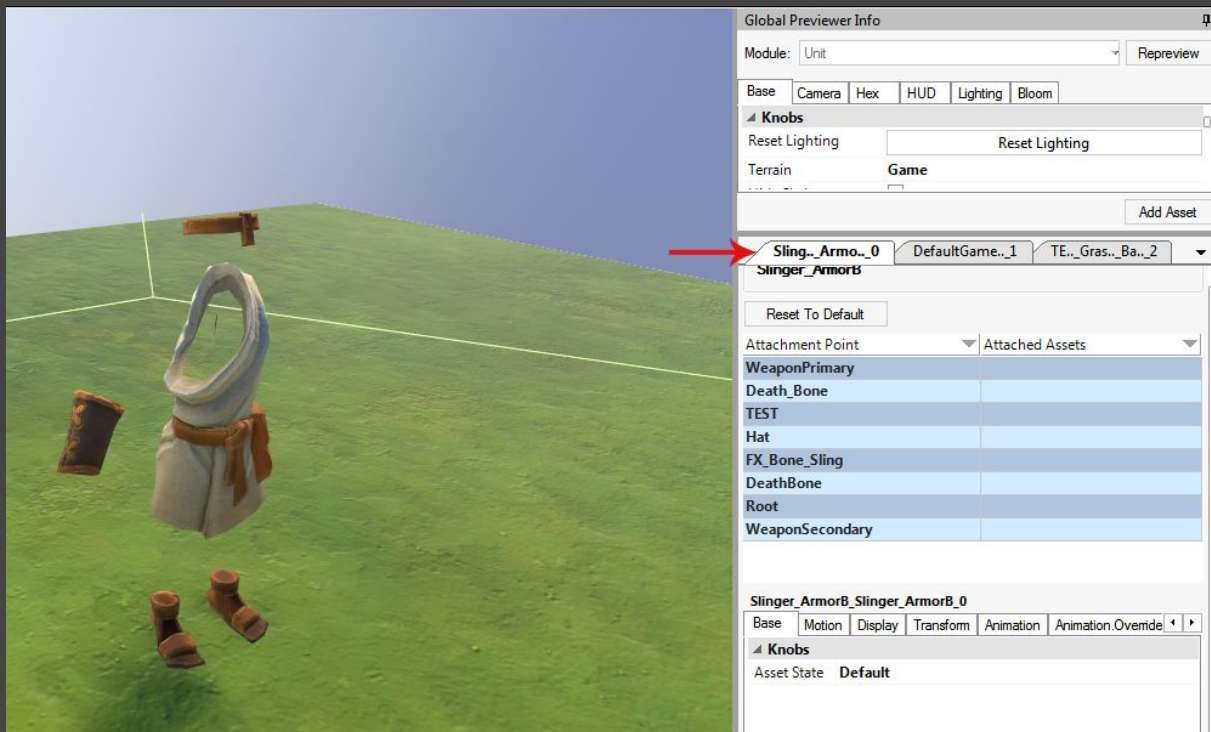
There's a lot of very useful stuff here. The Asset Previewer can help you check assets, rotate the camera by alt-clicking, add attachments, check animations, change tints, change lights, remove ground, and a lot more. We'll go through the basic options.

Putting attachments

Right now we just have the slinger armor, and that alone doesn't really look like a unit at all. So we need to add attachments!

Something important to note here. The Asset Previewer is just that, **a Previewer**, nothing we make here actually has any effect ingame. It is still a good tool to start since it can help visualize everything we'll need for our custom unit. This does mean however that all of these steps must be made in a single run; since closing the Asset Previewer will also remove whatever we do through it.

The Armor geometries come with different attachment points for accessories and bodies and etc. Most of the attachments will go in a point called "Root", but there are some specifics. In order to place attachments in the Asset Previewer you must find the tab corresponding our current asset.



From there you can click on the empty cells of the Attached Assets column to select attachments for every attachment point; you can place more than one attachment on each point. A window will appear in which you can search for existing assets to add. In this case we'll add the following:

- **WeaponPrimary:** Slinger_WeaponA
- **Hat:** WarCart_HelmA
- **Root:** Male_MiddleEastern_ThinBody_FullA, Horseman_Cloak A, Male_MiddleEastern_HeadO (yes, heads go in Root, not in Hat)

It's usually a good idea to write down the assets you're using somewhere, since we'll need this once we actually make the artdefine. Your result should be this:



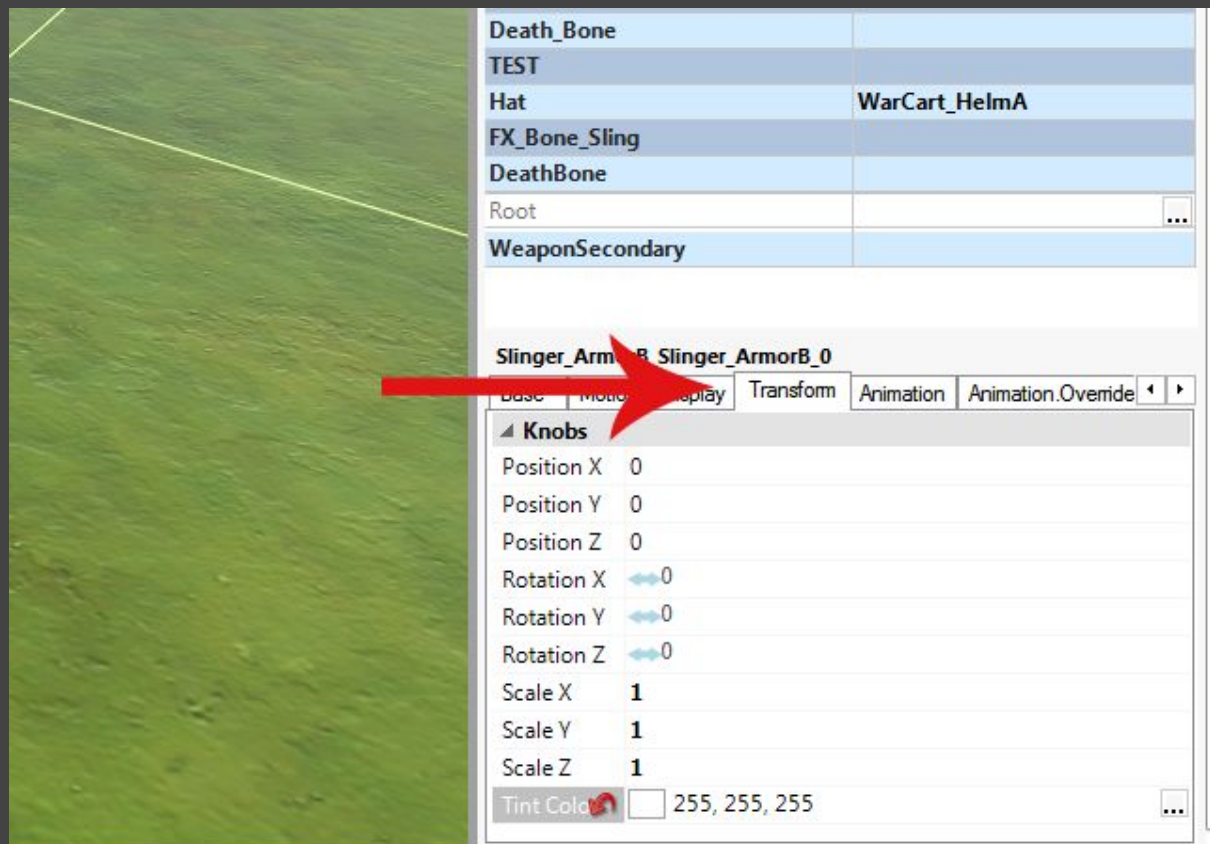
So, we now have something that almost looks like a custom unit! Cool right?

But we can make it look a bit more unique if we could change the white colour of the slinger (since all slingers have white clothing). Thankfully, there's a way to achieve this.

Tints in the previewer

So, there's something interesting in Civ 6 Assets, and is the fact that you can tint them to create colour variations without the need of custom textures.

In the case of our slinger guy we can do this. Check the Transform tab. There you'll find the Tint Color row.



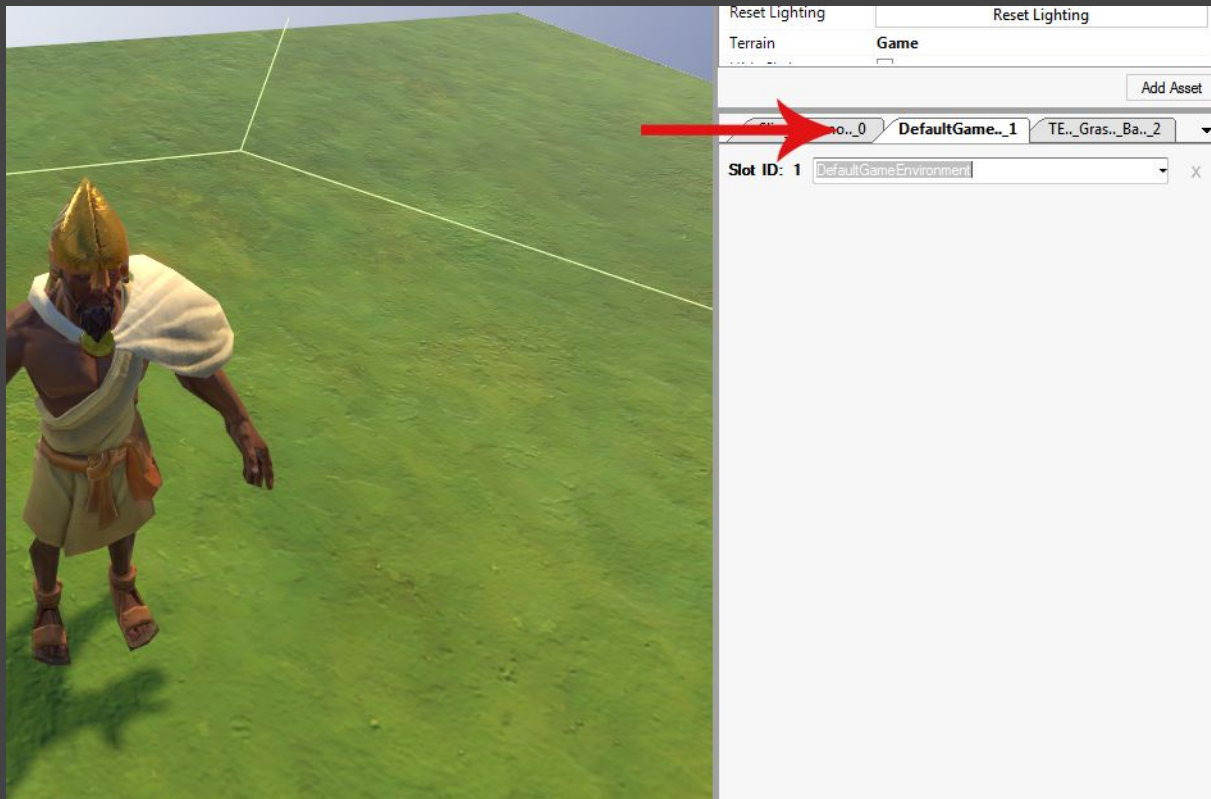
Click it and put the color: #DECF9C (222, 207, 156) or whatever colour you see fit. As you can see, the colour changed slightly.



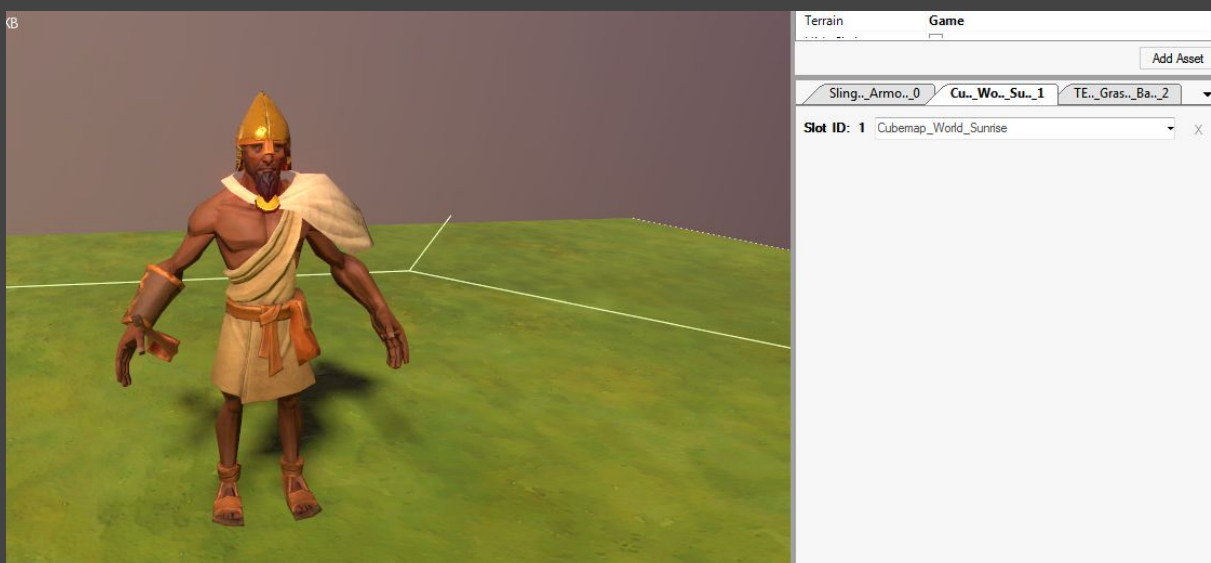
As you probably noticed, only the clothing changed, but the armor, skin and weapon all kept their original colours. This is because tints only apply to certain surfaces (this depends on how the material is set up on the original assets). So sometimes you might not be able to tint a part just as you wish.

Testing animations and lighting

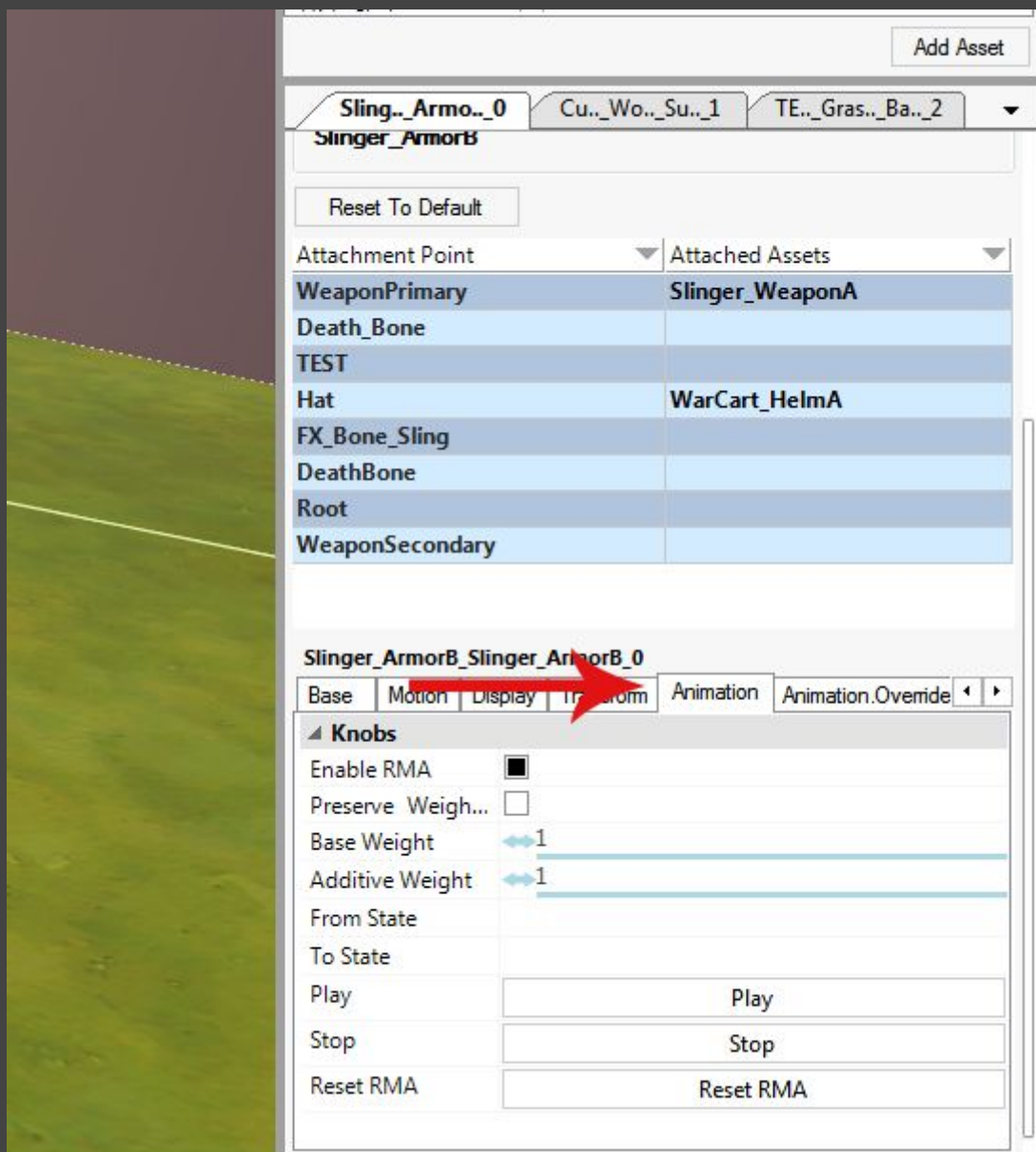
A nice thing of the Asset Previewer is that it lets you change the lighting of the scene you're looking at. While this is particularly useful for districts and landmarks, looking at units with different lights can be useful too. To change the lighting check the DefaultGameEnvironment Tab found next to your asset tab.



On the Slot ID 1 you can change to whatever you please. For this test we can use the “Cubemap_World_Sunrise” light.



Now something very important to do and remember is that we can test out animations in the Asset Previewer. This is very important since sometimes some parts contain specific animations (like other armors) and it’s good to see whether everything works in our setup. To test animations you can go to the Animation tab. (Go back to the Slinger_ArmorB tab if you can’t see it)



From there you can change the “To State” parameter and press play to test animations. Some animations will not play, but this is normal as not all models contain all stated variations for each attack or death. Try playing the ATTACK animation.



Don't worry if there's some strange clipping sometimes. From the game's perspective it really isn't noticeable. Anyway, it seems we got our unit moving. Now for the real work!

Defining the Artdefines

Checking the original artdefines

Now its the time to create our artdefines. Artdefs is how the game actually makes the connection between database unit and art. So we need to be very careful on this point.

I consider very important to check the original artdefines to follow a good example, specially on the first units one is making. It's also important since we have to know the UnitBin path of everything we are using.

.... Hold on... What the hell is a UnitBin?... Well, let's see. You saw how every asset we used on the previewer was kind of a different part right? UnitBins are technically folders (well, bins) where these parts are stored. For example, our body is on Bodies/Thin_Body. These paths are required when we make the artdefine.

There is a simple logic here, and it's to do everything in the right order:

UnitBins > UnitMembers > Unit

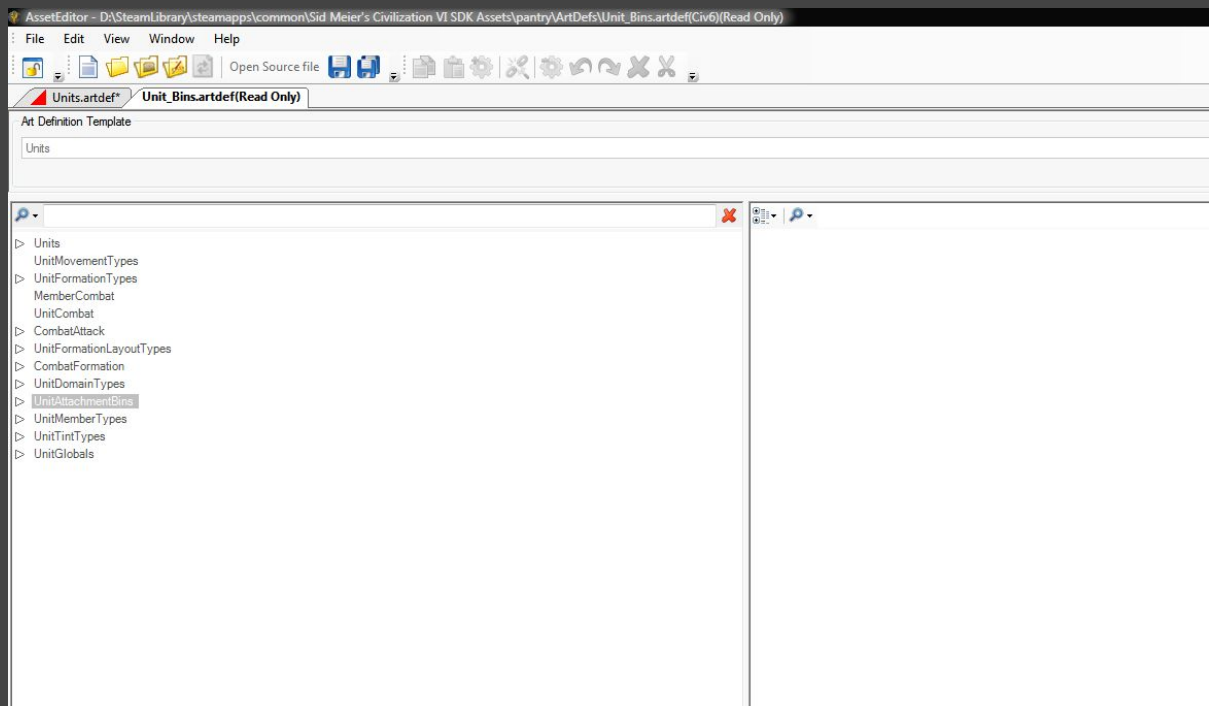
This means we'll first get our UnitBins for the assets we want; then we need to create the UnitMembers (which is every individual guy we have on the unit; usually 4 members in Civ6) and finally configure the art of the Unit itself.

UnitBins

So... since all of our assets are already in the base game, we can check the UnitBins from the pantry (you must have downloaded the art files of course). You can find the pantry and the ArtDefines here:

[SteamLibrary]\steamapps\common\Sid Meier's Civilization VI SDK Assets\pantry\ArtDefs

We'll need to open the Unit_Bins.artdef file using the Asset Editor. Click on File>Open Art Def to do this. You should see something like this:



Expand the UnitAttachmentBins category and you'll find a bunch of different folders (Bins) with generic names like Accessories or Armor. What we need to do now is search for our assets in the bins.

Bins have a simple structure:

Bin Name > Groups > Group Name > Cultures > CultureName > Assets

Bin Name defines the Category an asset is part of (Like heads or Spears or whatever). The Group points to the actual name that will be referenced later in the artdefine (Like Slinger, or NorwegianBerzerker) ... Cultures is of course a reference to cultural variants (like skin colours or helmets) and finally you have the Assets. If a Group has more than one asset they'll be chosen randomly (like head variations for example).

So, look around and search for our required Bins.

...
...
...

Alright, these are the bins we need:

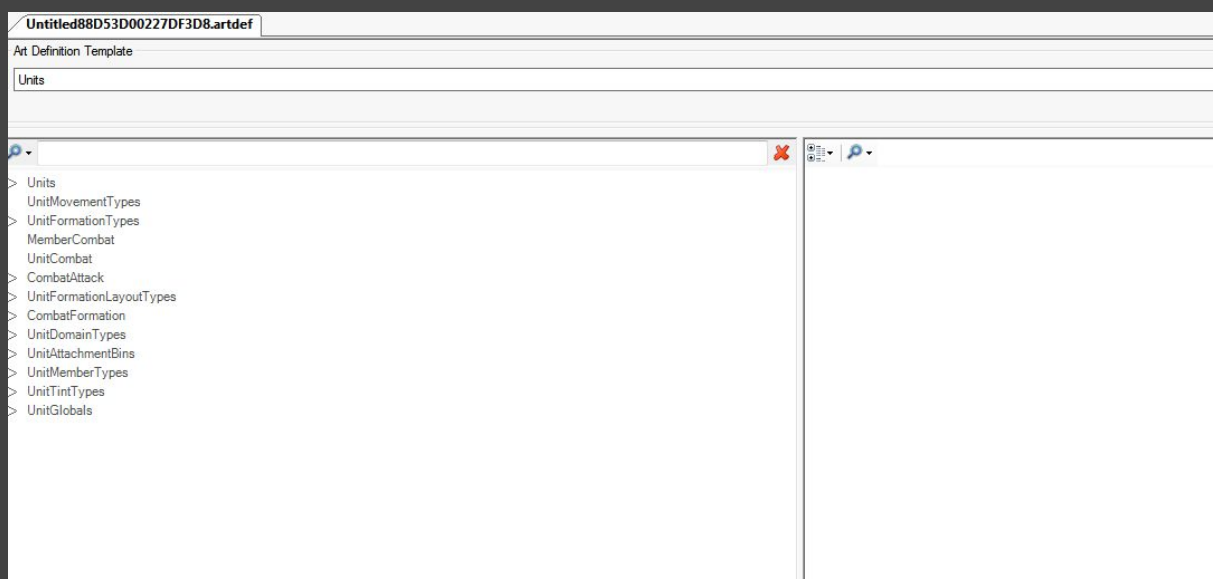
Weapons/Slinger
Hats/WarCart
Armor/Slinger
Bodies/Male_ThinBody_Full
Capes/Horseman
Heads/Male

As you can see, for the Bin paths we only need to know the Bin Name and the Group Name. The cultural data is just for variations so we won't need it (However, if we want a specific asset that isn't alone in a group we'll have to create a new Bin... more on that on the next level of course).

Tints

Now, remember that we used a Tint on our unit at the preview? We need to add that tint to use it in the game too.

At the Asset Editor go to File > New. Select ArtDef from the Packages category. It will likely open the Units Template, but if it doesn't select "Units" from the dropdown menu at Art Definition Template. You should have something that looks like this:



Before we proceed, save the Artdef file. You can put it whatever name you wish, but for this exercise we'll call it "Units.artdef" ... Remember to save it on the Artdefs folder of your Project please!

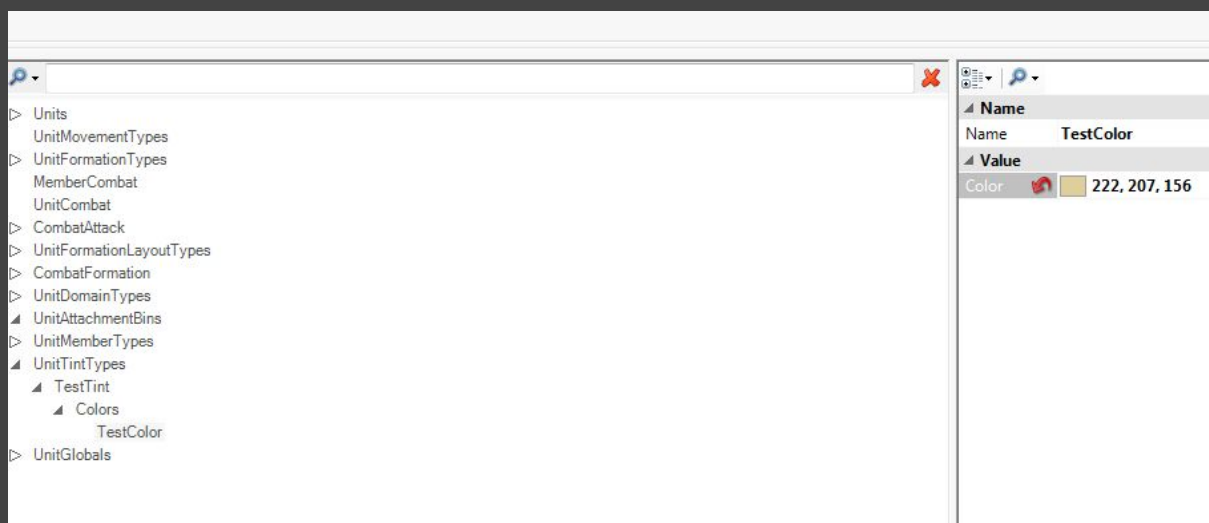
Now, on the UnitTintTypes category, right click and select "Add Element". Expand and on the Colors subcategory also add an element. Expanded you should see something like this:

UnitTintTypes > UnitTintTypes > Colors > Colors

Seems rather redundant right? But it's only because we haven't changed the name of our Tint. To change the name you must edit the cell that says "Name" on the right side of the screen. You can use whatever name you want but we'll go with TestTint. The Colors can also be named anything so we'll call it TestColor. This should be your hierarchy now:

UnitTintTypes > TestTint > Colors > TestColor

If you select TestColor you'll see a cell on the right that will allow you to actually change the color (right now should be set at White 255, 255, 255). Click on it and then paste the colours we used before on the preview: 222, 207, 156 . It should look like this:

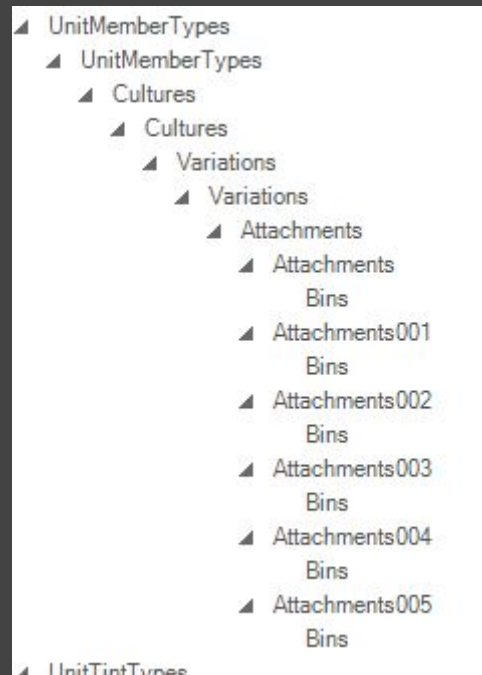


Now that we have our tint defined we can proceed with the Unit Member.

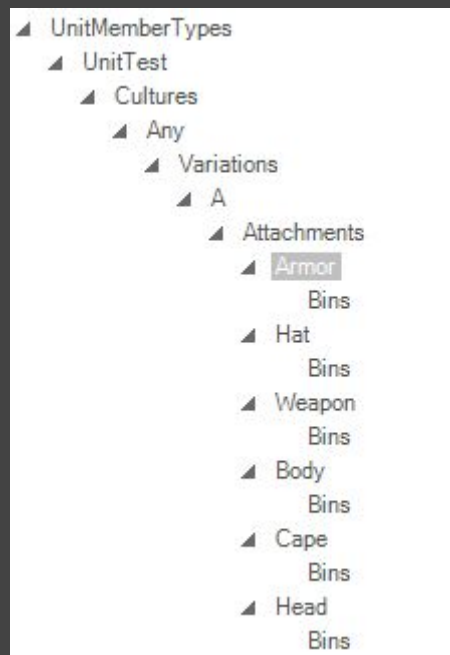
UnitMember

Just as we created the tint, we can use Add Element to add a UnitMemberType. Add Elements on the UnitMemberTypes category, expand it and add on the Cultures category, then expand and add on the Variations category, and finally expand and add 6 times on the Attachments category.

If that was confusing, all I mean is you should add elements until you get a hierarchy like this:



Now we proceed to renaming each subcategory as we see fit, following the process we made when we were defining the Tint. You should have this hierarchy:



UnitTest is the name of our UnitMember. We use Any at Cultures since we won't be needing cultural variations for the unit (usually the case for UUs). And we only want to make a single variation, which we'll call A but there isn't really any importance to it.

Now that we have the setup complete we can actually start filling those spots. Usually it's good to follow a base from the original files (Units.artdef on the pantry). In every single of these subcategories you'll have to fill in some blanks that will define how your unit will look so it's very important to not miss anything. From these point we'll just grab stuff from the original artdefine, but most of it should be rather self-explanatory.

So... Under UnitTest:

Name	
Name	UnitTest
Value	
Movement	Slinger
Combat	SlingerCombat
VFXMaterialType	MEAT
VFXWeaponImpact	MEAT
ImpactHeightOverride	0

Under variation A:

Name	
Name	A
Value	
Scale	1,1
New Parameter	
IsAttachment	<input type="checkbox"/>

Note: Most ground non-mounted units use a scale of 1,1 ingame.

Under attachment Armor

Name	
Name	Armor
Value	
Point	Root
Tint	TestTint

Note: If you made the Tint setup correctly you should be able of just typing TestTint as the name without any issues. We'll also need to add the UnitBin to this attachment. To do this add an Element on the "Bins" subcategory of Armor. Nothing will appear after it, but on the right side of the screen you should see an empty cell under a Name column. Fill it with the path we wrote before for Armor:

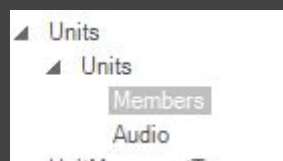
Name
Armor/Slinger

Repeat the last one with the other attachments [Hat, Weapon, Body, Cape, Head]
(remember we wrote it down while we were previewing, everything goes on Root except for the weapon which goes on WeaponPrimary and the helmet which goes on Hat).

Once you do that, save the .artdef file; and we can finally add the actual Unit define.

Unit ArtDefine

Following the same process as before, add elements under the Units category until you have this hierarchy:

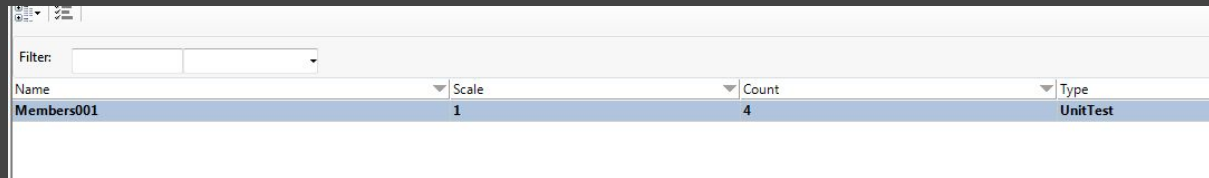


Also add a single Element under Members and Audio. Change the name of the Units subcategory to UNIT_TEST (this is how the game realizes this artdefine is for our actual unit). And inside it fill this data:

Name	
Name	UNIT_TEST
Value	
Formation	Slinger
UnitCombat	Slinger
EscortFormation	WarriorEscort
EmbarkedUnit	UNIT_ANCIENTEMBARK
DoNotDisplayCharges	<input type="checkbox"/>
Culture	
Era	
ProxyUnit	
PlayDeathOnDestroy	<input type="checkbox"/>
DisplayLevel	<input type="range" value="0"/>

Then under members you should see 4 Columns: Name, Scale, Count and Type. Sometimes the AssetEditor shows only the Name column, this is either fixed by opening the original

Units.artdef or trying to change the size of the columns (by changing the size of the window or by scrolling on the right side on the Name column to change its size). On the cells fill this data:

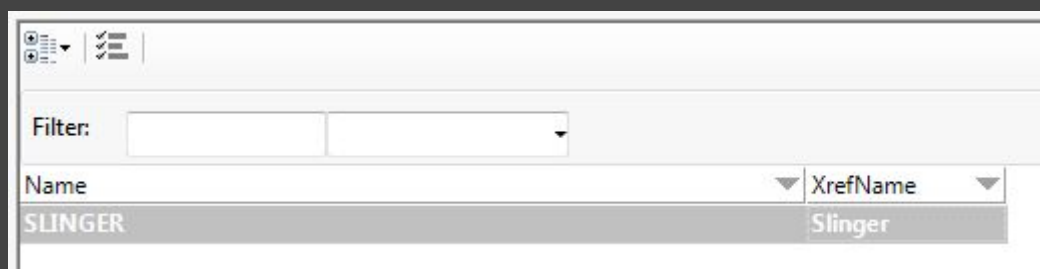


The screenshot shows a table editor with a 'Filter' section at the top. Below it is a table with four columns: 'Name', 'Scale', 'Count', and 'Type'. The first row of data is highlighted in blue and contains the values 'Members001', '1', '4', and 'UnitTest'.

Name	Scale	Count	Type
Members001	1	4	UnitTest

If we did everything correctly when defining the UnitMember, filling the Type with UnitTest should work with no issues.

Under Audio you should fill this information:



The screenshot shows a table editor with a 'Filter' section at the top. Below it is a table with two columns: 'Name' and 'XrefName'. The first row of data is highlighted in grey and contains the values 'SLINGER' and 'Slinger'.

Name	XrefName
SLINGER	Slinger

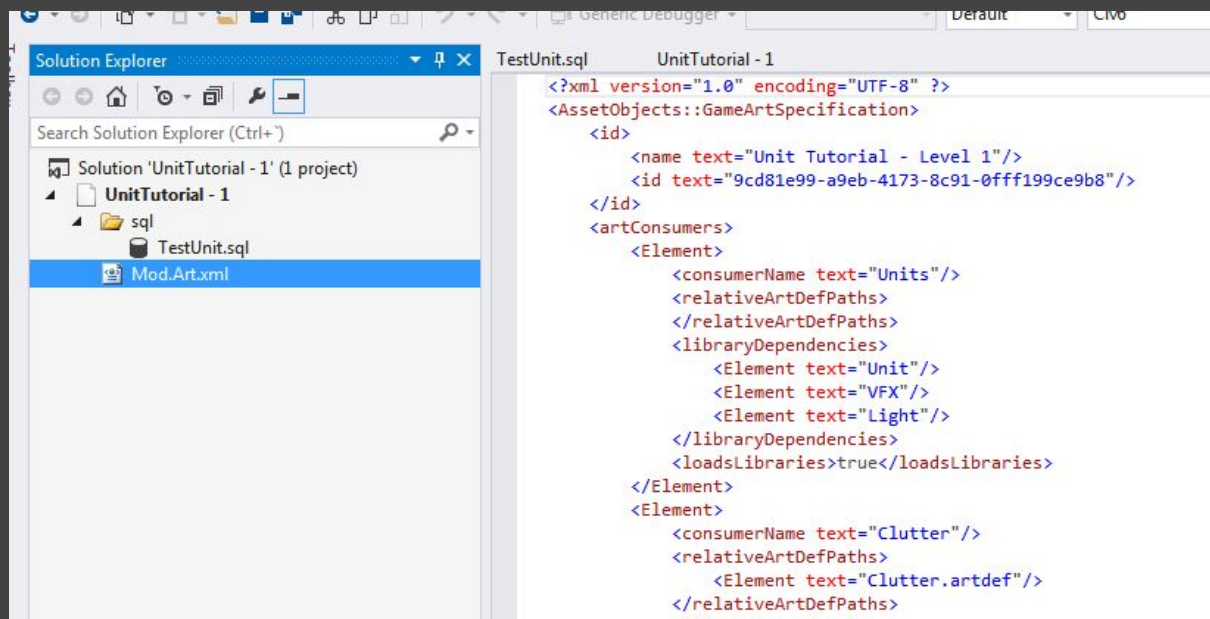
And with that our ArtDefine should be complete! Congratulations! We're almost done

The ModArt File

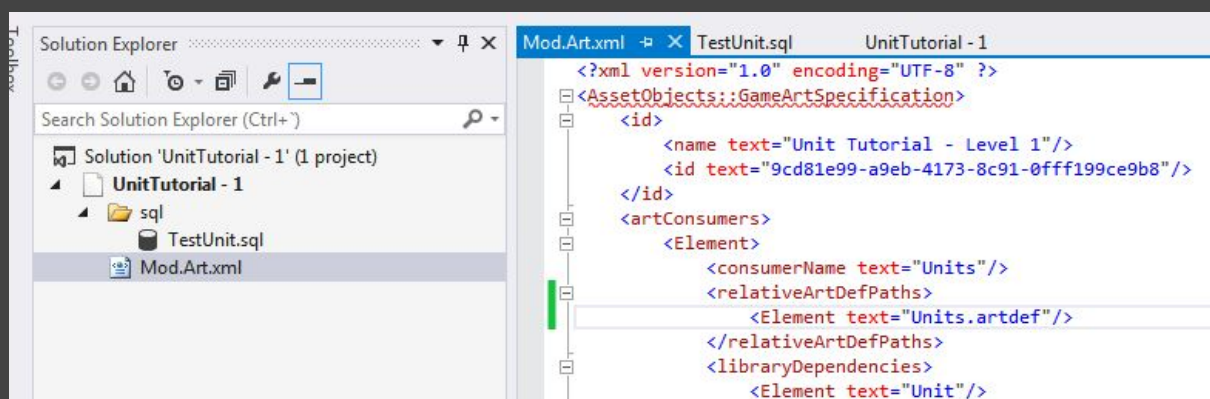
You might feel tempted to test the artdefine right away, but there is a step many modders forget and it's setting up everything on the Mod.Art.xml file.

Every mod project comes with a mod.art.xml file (and if yours doesn't you probably picked the wrong template when creating your mod). In theory this file auto-updates so it can cook the artdefines and xtps into data the game can actually read. Most of the times one has to update it manually. The good news is we can finally close the Asset Editor.

If you explore your solution with ModBuddy you should see a the file, open it and it'll look like this:

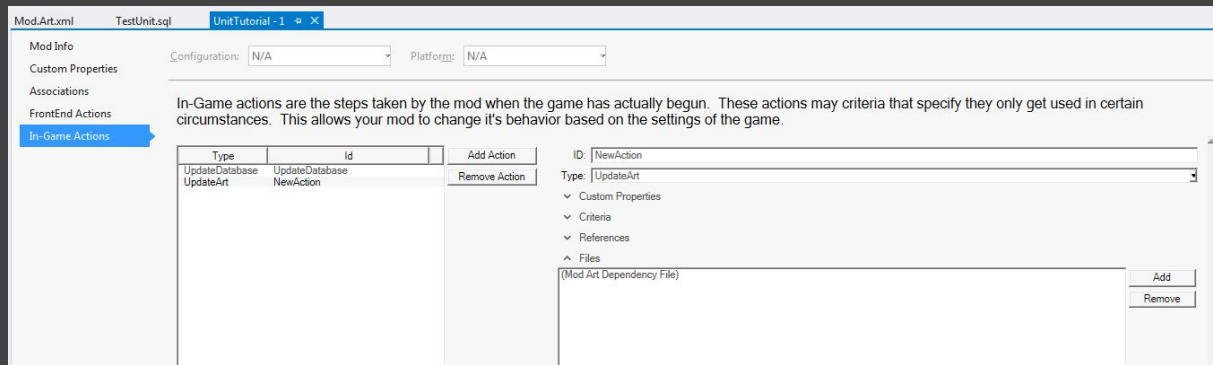


The mod.art.xml file is made of artConsumers which are the ones that should be fed (so to speak) with the ArtDefs we made. As you can see there, the Units consumer doesn't have any relativeArtDefPath. Fix that by adding our ArtDefine file (which we conveniently called Units.artdef)



Always remember to do this with every single artdef you are adding to your mod. Otherwise it will just not be read and all will be for naught.

Another thing you must make sure is that the Modart file is correctly set up in the Mod Properties (If you don't know how to access the mod properties, learning how to make a unit isn't really your main issue). You should have something like this:



Now save the files and we can finally build our mod and test it!

Test time!

Remember to have the Unit Test mod active. You can use the Firetuner or train the unit to test it ingame. You should see it like this:



