Civ 6 Unit Asset Tutorials
Level 2 - Change your behavior!

By Leugi

Mixing and tinting ingame assets is usually enough for making Civ6 units, but sometimes you will meet up with some issues. If you wanted to use, say, the cavalry armor for a melee unit, you can't directly because of animations. The managing of mounted units is also slightly different to that of unmounted ones, so we'll be covering that up too. And of course, units need icons, so that's kind of an issue right?

So we'll now explore that, as well as some basic but very necessary stuff about .XLPs

This tutorial assumes you know everything from Level 1, so if you don't know how to set up an unit's artdefine or what the hell the modart file is I must tell you to go back to that.

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 behavior!:** Introduces creating custom assets, changing the animations, adding xlps, 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.

What you need:

- Civ 6 and Civ6 SDK Tools.
- Patience. Required to run the Asset Editor.
- This base project file. (includes a dummy unit for testing purposes)

Creating our base Asset

Save As

So we need to start by opening the Asset Editor and the Asset Previewer, come back in like 2 hours when that opens…

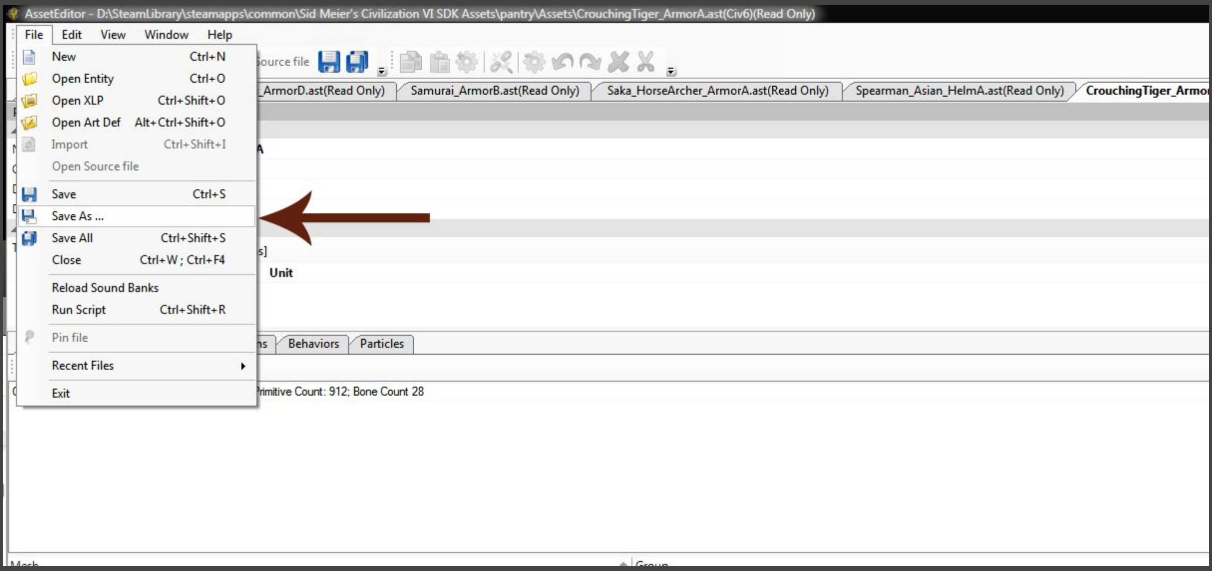
…

Ok good, now we can start. We are going to make an interesting mounted unit so let's use the cavalry as a base. We're going to pick an example that will not only look nice, but also has a lot of the worst things that you could find when changing behaviors (it’s not that bad tho). Open up the CrouchingTiger_ArmorA asset and preview it.

ws: 31
terized Tris: 2542
atistics ***
Instances: 912
t Instances: 792
erenced Geometry Data: 30.09 KB
erenced Texture Data: 1066.65 KB
erenced Animation Data: 1949.77 KB
Refreshed At: 7:08:08 am
Of Day: Day

Cool huh? However, this armor is not easy to use for custom units for a single reason. Its animations are that of a guy that uses a cannon. Thankfully there’s a way to fix this, so we’ll give him a horse!

The first thing we have to do is creating a new asset based on the Crouching Tiger one. To do this go to File > Save As.


AssetEditor - D:\SteamLibrary\steamapps\common\Sid Meier's Civilization VI SDK Assets\pantry\Assets\CrouchingTiger_ArmorA.ast(Civ6)(Read Only)
File
Edit
View
Window
Help
New Ctrl+N
Open Entity Ctrl+O
Open XLP Ctrl+Shift+O
Open Art Def Alt+Ctrl+Shift+O
Import Ctrl+Shift+I
Open Source file
Save Ctrl+S
Save As ...
Save All Ctrl+Shift+S
Close Ctrl+W ; Ctrl+F4
Reload Sound Banks
Run Script Ctrl+Shift+R
Pin file
Recent Files
Exit
_ArmorD.ast(Read Only)
Samurai_ArmorB.ast(Read Only)
Saka_HorseArcher_ArmorA.ast(Read Only)
Spearman_Asian_HelmA.ast(Read Only)
CrouchingTiger_Armo
Unit
Behaviors
Particles
Primitive Count: 912; Bone Count 28

A popup will appear asking you for a new name for your entity. **It is very important to give it the right name here,** since changing names afterwards can be rather buggy and krill your

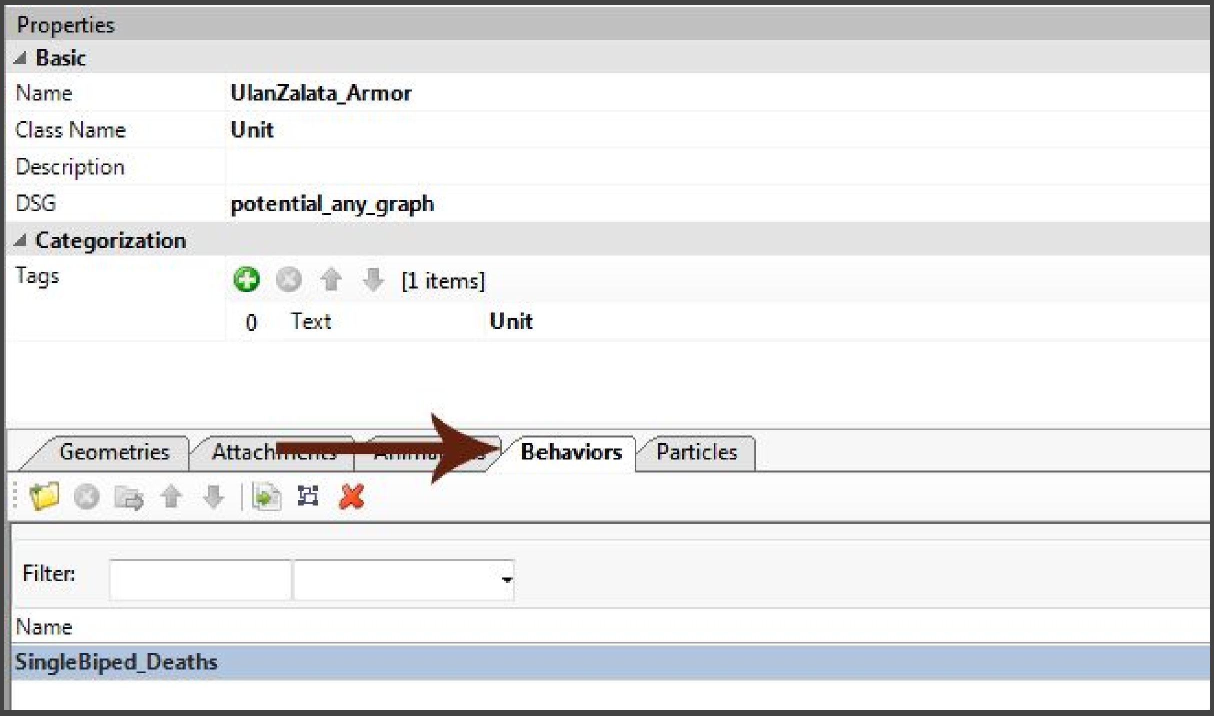
Asset Editor! So we'll name it UlanZalata_Armor. After a bit your file should be saved and your new asset is created.

Changing Behaviors

So, how do we change the animations to turn this cute model into a horseman? Well, animations in civ6 are handled through what we call "Behaviors".

Think of Behaviors as a kind of package that contains animations for specific unit types. So for example a "Horseman" Behavior will give the model the animations we need.

So, to check Behaviors go to the "Behaviors" tab on the Asset Editor.


Properties

Basic

Name	UlanZalata_Armor
Class Name	Unit
Description	
DSG	potential_any_graph

Categorization

Tags	[1 items]		
	0	Text	Unit

SingleBiped_Deaths

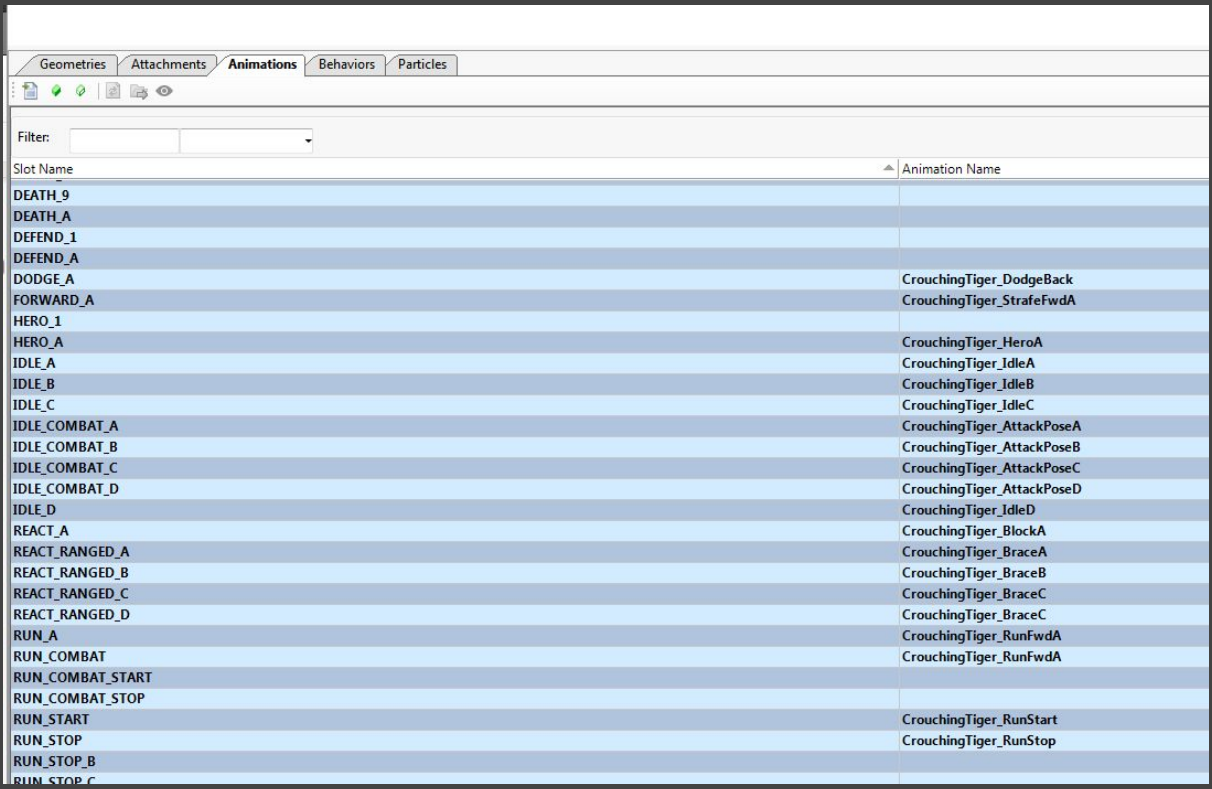
First erase the SingleBiped_Deaths (we need another one for a Mounted Biped) And then you need to load the following Behaviors:

- Horseman
- SingleBiped_MountedDeaths

Save your Asset. And try to preview an animation. Unless you preview the Death animation, you'll notice that nothing has changed D:

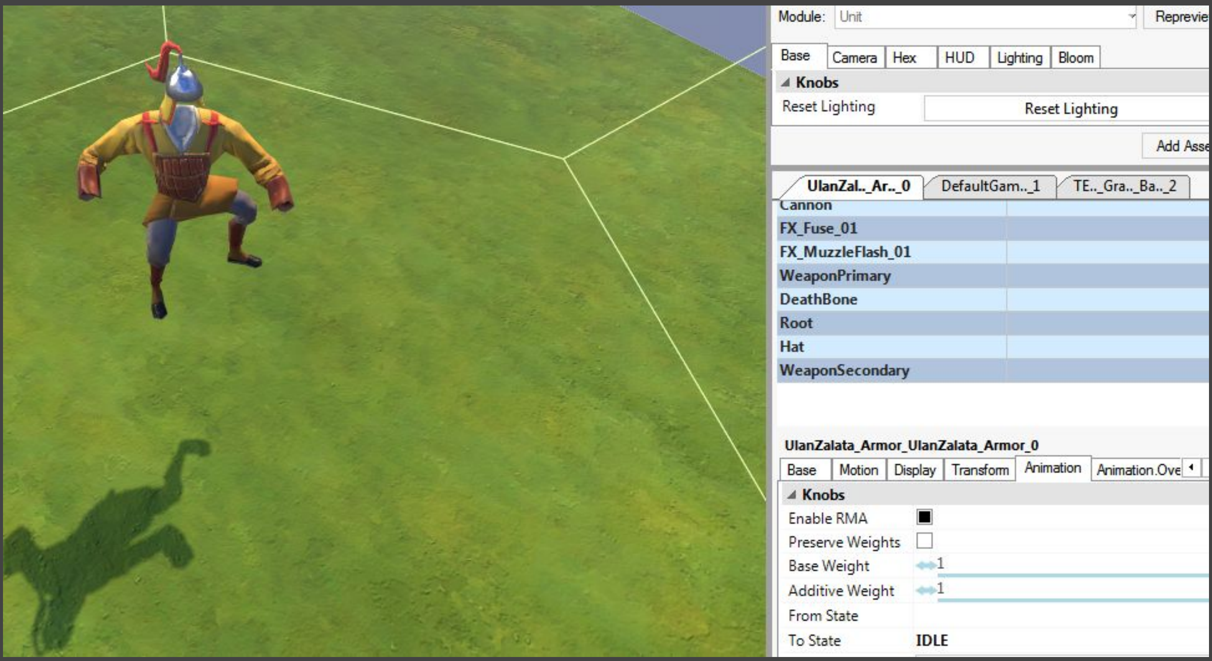
This is because the Crouching Tiger is a very special case. Aside from Behaviors, some models can have specific animations that will override the behaviors. We'll need to remove these ones if we want to have him mounted. Go to the Animations tab next to Behaviors.

There you will see a list of all Animations, and then a row called “Animation Name”. This is where the overridden animations come from.



Slot Name	Animation Name
DEATH_9	
DEATH_A	
DEFEND_1	
DEFEND_A	
DODGE_A	CrouchingTiger_DodgeBack
FORWARD_A	CrouchingTiger_StrafeFwdA
HERO_1	
HERO_A	CrouchingTiger_HeroA
IDLE_A	CrouchingTiger_IdleA
IDLE_B	CrouchingTiger_IdleB
IDLE_C	CrouchingTiger_IdleC
IDLE_COMBAT_A	CrouchingTiger_AttackPoseA
IDLE_COMBAT_B	CrouchingTiger_AttackPoseB
IDLE_COMBAT_C	CrouchingTiger_AttackPoseC
IDLE_COMBAT_D	CrouchingTiger_AttackPoseD
IDLE_D	CrouchingTiger_IdleD
REACT_A	CrouchingTiger_BlockA
REACT_RANGED_A	CrouchingTiger_BraceA
REACT_RANGED_B	CrouchingTiger_BraceB
REACT_RANGED_C	CrouchingTiger_BraceC
REACT_RANGED_D	CrouchingTiger_BraceC
RUN_A	CrouchingTiger_RunFwdA
RUN_COMBAT	CrouchingTiger_RunFwdA
RUN_COMBAT_START	
RUN_COMBAT_STOP	
RUN_START	CrouchingTiger_RunStart
RUN_STOP	CrouchingTiger_RunStop
RUN_STOP_B	
RUN_STOP_C	

Delete everything that is on Animation Name. Save and go to preview an animation. If you did everything well, it should now have actual Horseman animations!

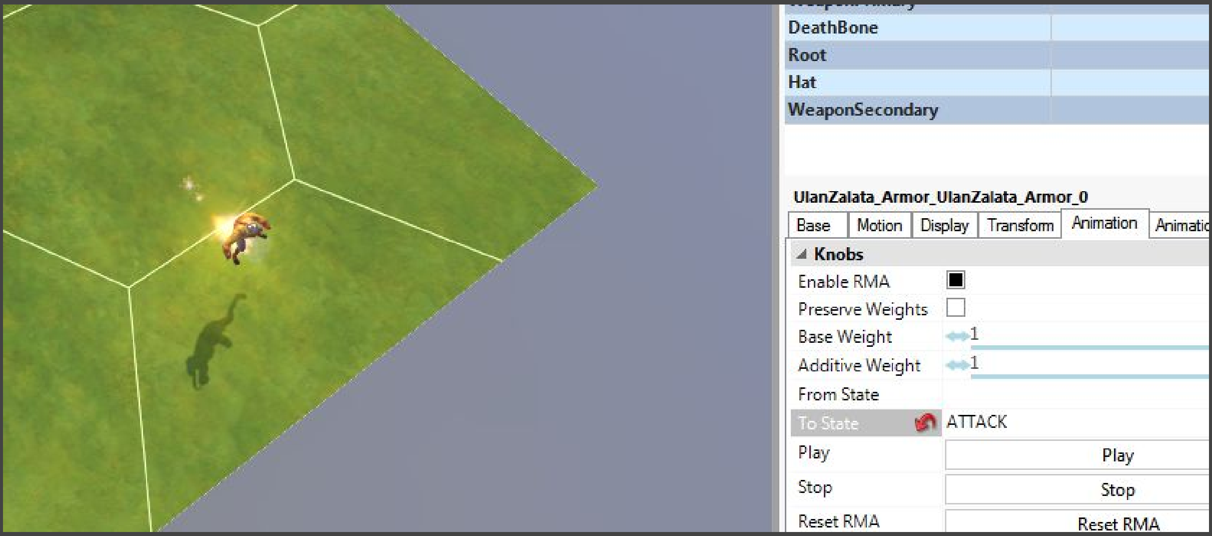


Cannon
FX_Fuse_01
FX_MuzzleFlash_01
WeaponPrimary
DeathBone
Root
Hat
WeaponSecondary

UlanZalata_Armor_UlanZalata_Armor_0

Enable RMA	■
Preserve Weights	☐
Base Weight	1
Additive Weight	1
From State	
To State	IDLE

Now… Zoom out your preview and let's see the Attack animation! (Zooming out is useful because Horsemen move a lot when attacking)



UlanZalata_Armor_UlanZalata_Armor_0

Enable RMA	■
Preserve Weights	☐
Base Weight	1
Additive Weight	1
From State	
To State	ATTACK
Play	Play
Stop	Stop
Reset RMA	Reset RMA

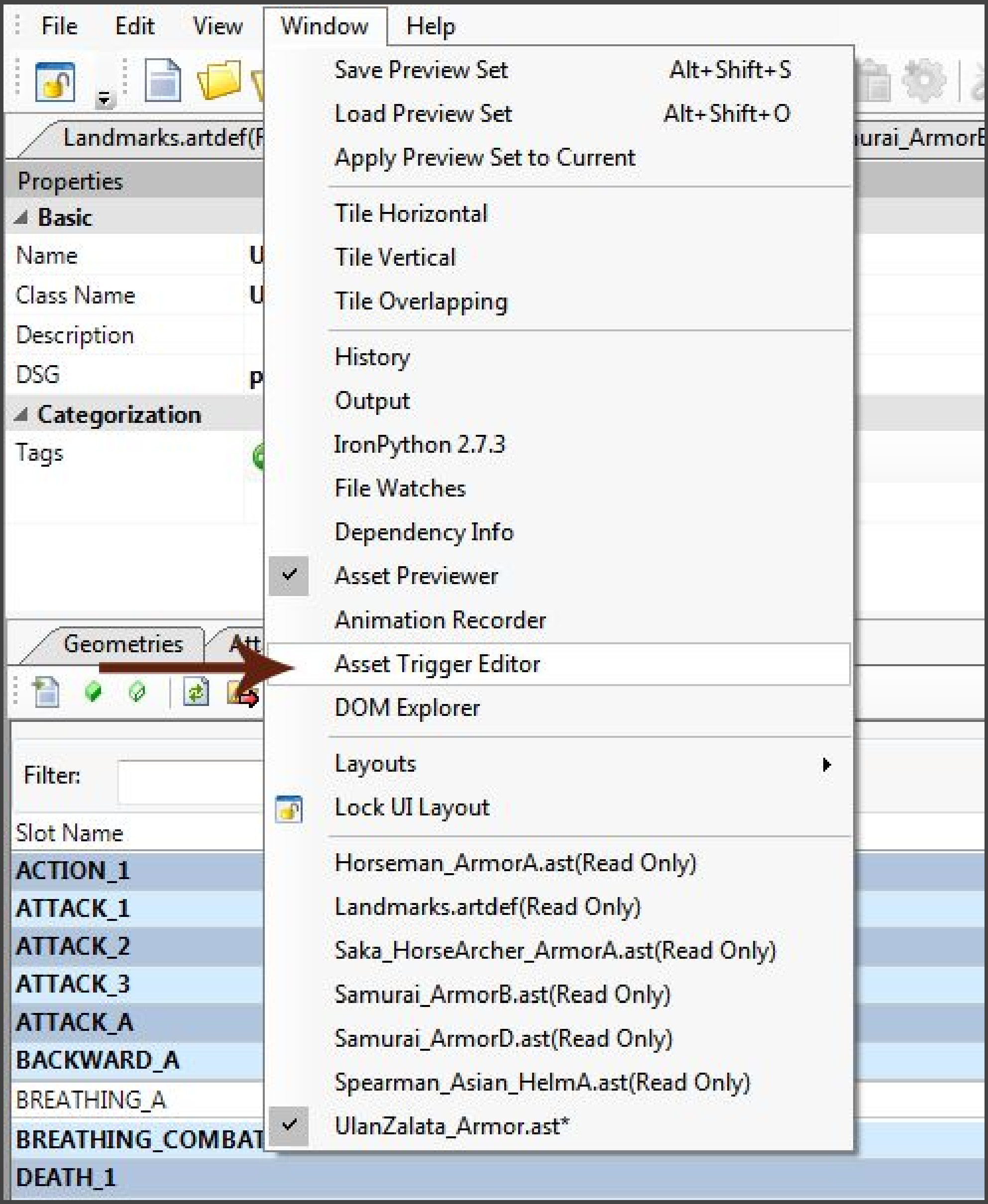
If you managed to pay attention, our attack animation is weird. It has this strange glow, as if there was some sort of fire or something. This is because the Fire effect from the Crouching Tiger is still there (since they attack with actual cannons) *[BTW, if you want to reposition the asset back to its origin point you can click on the Repreview button]*

So… what do we do now?

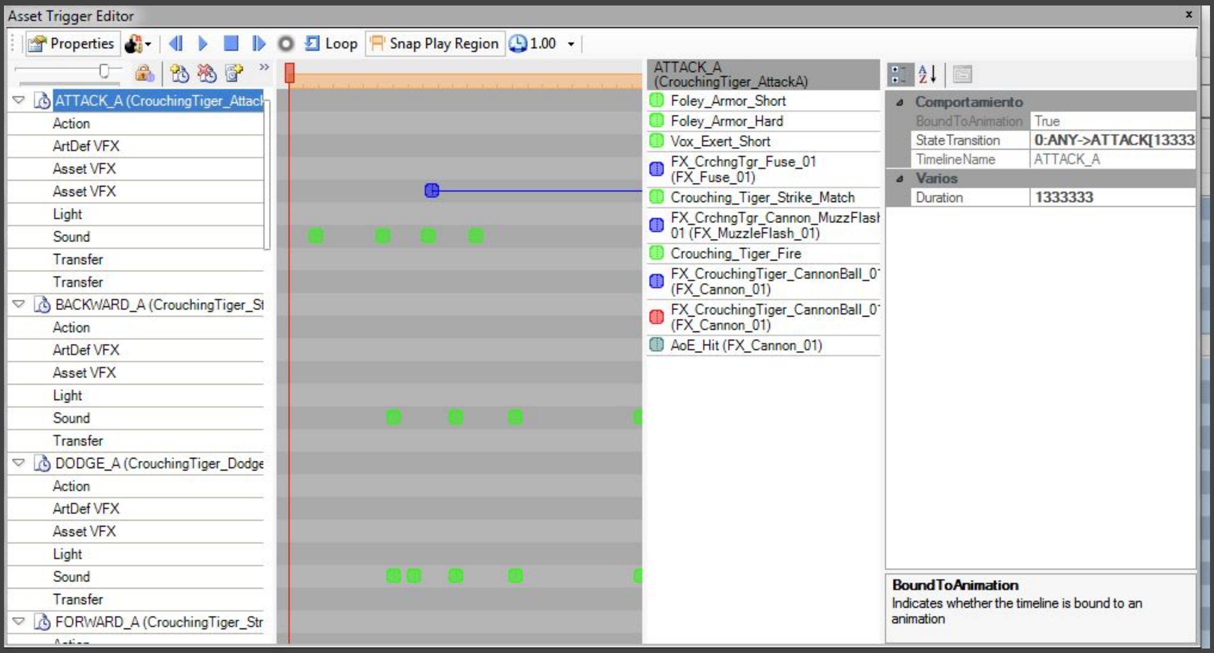
Triggered

Assets have something interesting called Triggers. Triggers are how certain Visual FX and Sound FX are set to run at specific times during specific animations; this is how the game tells when to put that shiny light on the cannon for the Crouching Tiger.

In order to modify triggers of a given asset we need to access the Asset Trigger Editor, right on the Window Menu.



Once open you should see something like this:



ATTACK_A
(CrouchingTiger_AttackA)
- Foley_Armor_Short
- Foley_Armor_Hard
- Vox_Exert_Short
- FX_CrchngTgr_Fuse_01 (FX_Fuse_01)
- Crouching_Tiger_Strike_Match
- FX_CrchngTgr_Cannon_MuzzFlash 01 (FX_MuzzleFlash_01)
- Crouching_Tiger_Fire
- FX_CrouchingTiger_CannonBall_0 (FX_Cannon_01)
- FX_CrouchingTiger_CannonBall_0 (FX_Cannon_01)
- AoE_Hit (FX_Cannon_01)

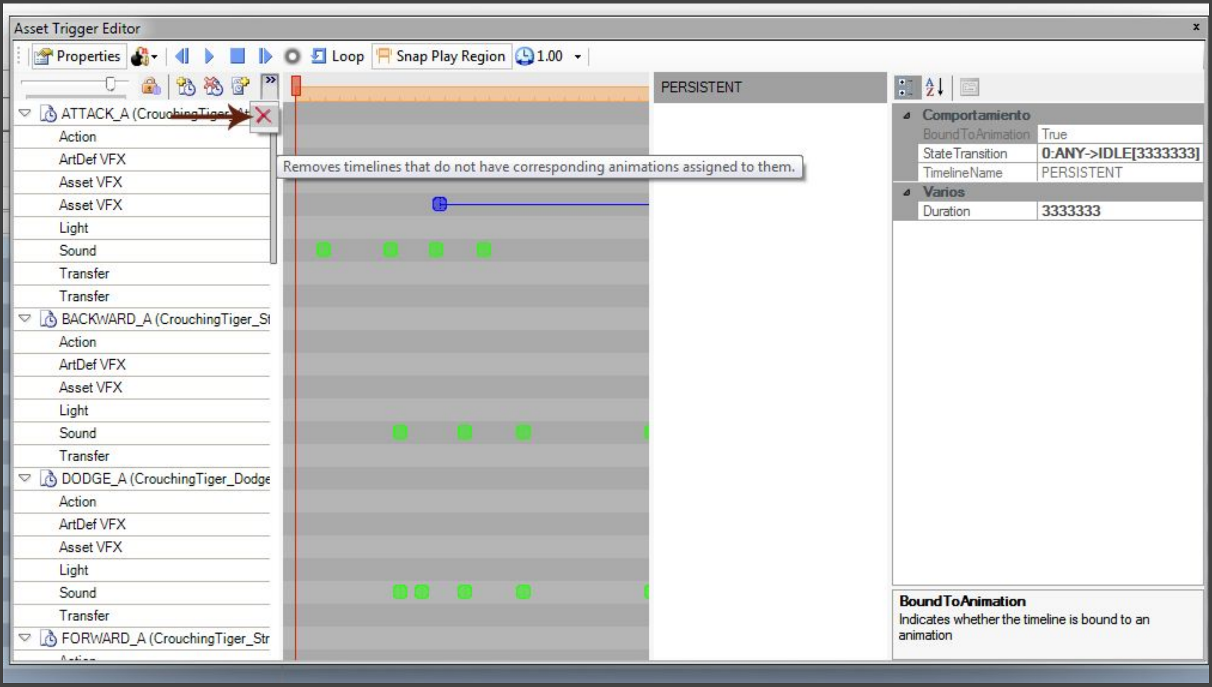
Comportamiento	
BoundToAnimation	True
StateTransition	0:ANY->ATTACK[13333
TimelineName	ATTACK_A
Varios	
Duration	1333333

BoundToAnimation
Indicates whether the timeline is bound to an animation

Very creepy right? We won’t go into full detail for the purposes of this tutorial, but as you can see there’s a bunch of VFX, light and sound definitions for different animations, as well as a timeline thing with keys set to it.

In most cases these Triggers are set on the Behaviors (so, the horseman part of it is already in now that we set it as the behavior), this means that we can just get our way by removing all Triggers from this asset. *(phew)* If you ever need to add particle effects or fire to a unit this could be handy though (for example a person holding a torch)

There is a hidden button that Removes Timelines that don’t have an animation assigned to them, we’ll use this one first for cleaning up.

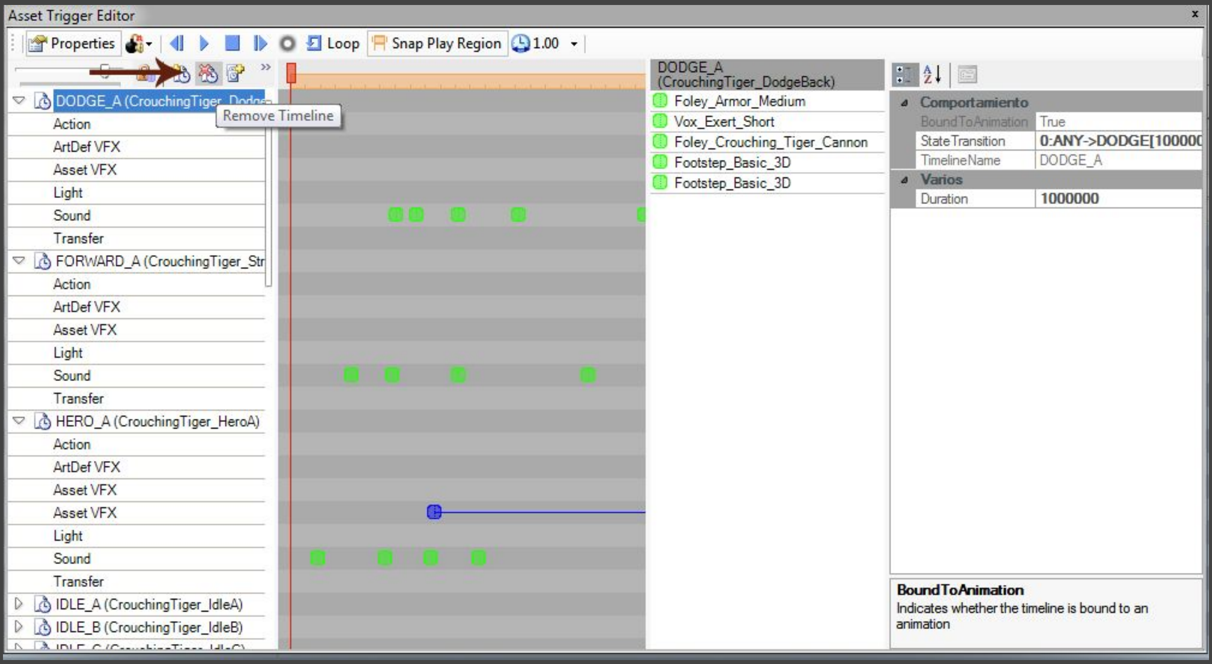


Removes timelines that do not have corresponding animations assigned to them.

Comportamiento	
BoundToAnimation	True
StateTransition	0:ANY->IDLE[3333333]
TimelineName	PERSISTENT
Varios	
Duration	3333333

BoundToAnimation
Indicates whether the timeline is bound to an animation

If any animation still exists after that, select an animation name and press the Remove Timeline button to kill it.



DODGE_A
(CrouchingTiger_DodgeBack)
- Foley_Armor_Medium
- Vox_Exert_Short
- Foley_Crouching_Tiger_Cannon
- Footstep_Basic_3D
- Footstep_Basic_3D

Comportamiento	
BoundToAnimation	True
StateTransition	0:ANY->DODGE[100000
TimelineName	DODGE_A
Varios	
Duration	1000000

BoundToAnimation
Indicates whether the timeline is bound to an animation

You can now close the Asset Trigger Editor, save your asset and rejoice because all traces of cannon shooting are gone!

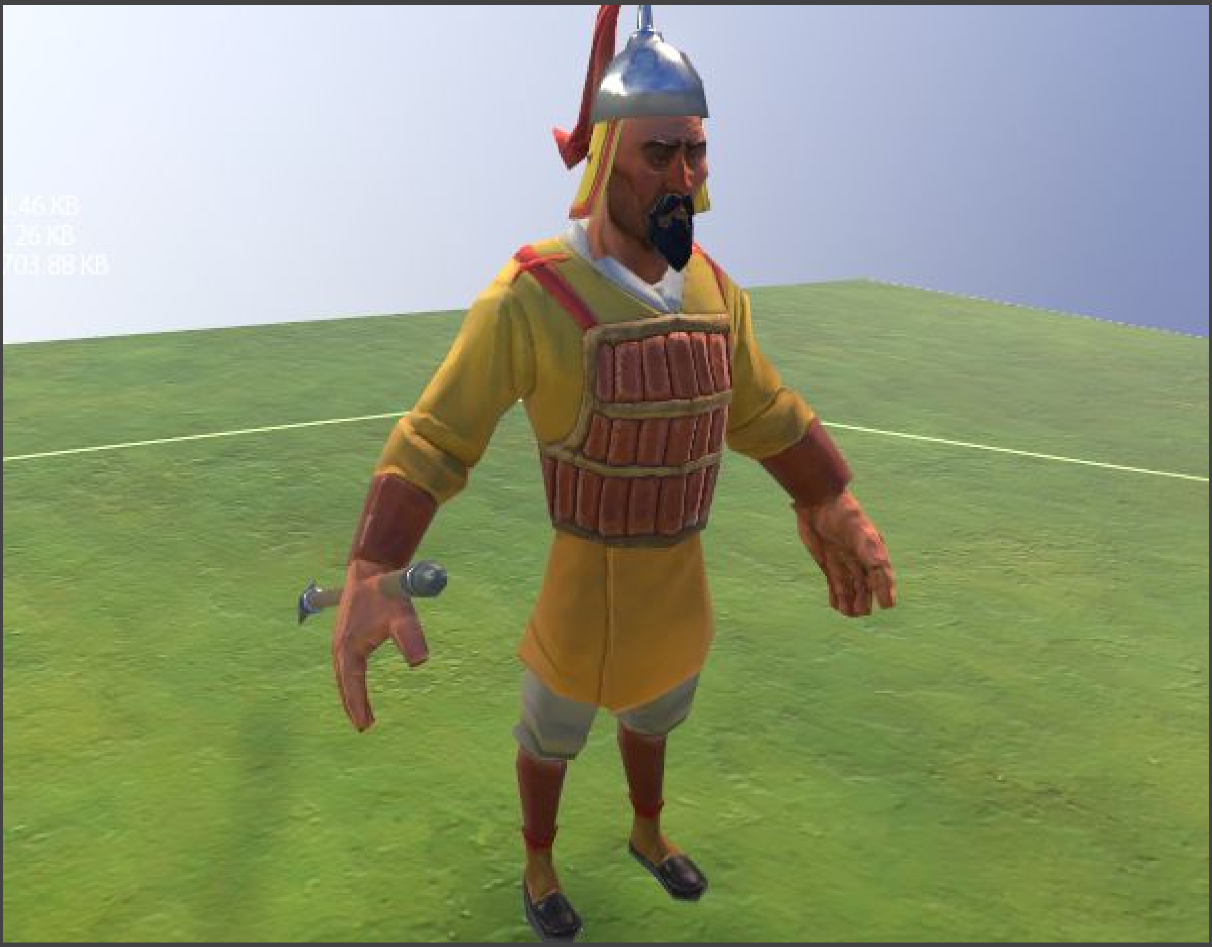
(BTW, in most cases you won't have to touch the Asset Trigger Editor; just when the asset includes it apart from the behaviours)

Custom Attachments

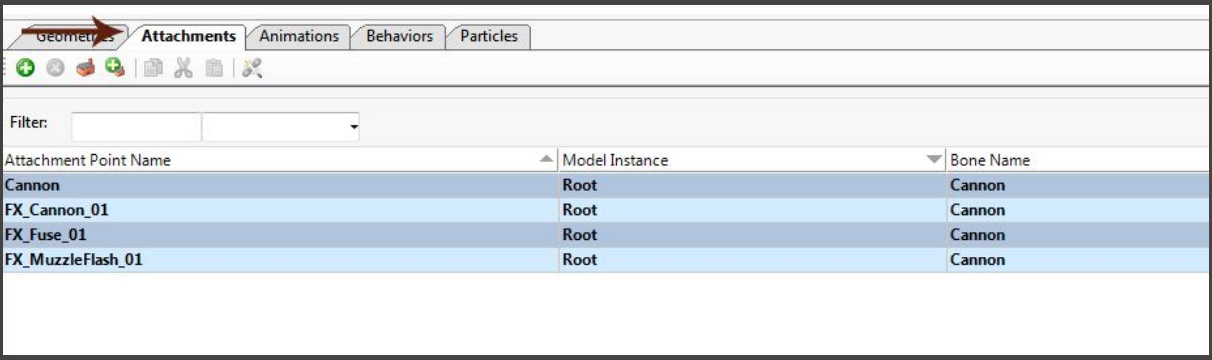
Alright let’s set up our rider on the preview. Reset the preview and stop all animations. Put the following assets on the following attachment points:

- **Root:** Male_Asian_MediumBody_HandsA, Male_Asian_HeadF
- **WeaponPrimary:** Horseman_SpearA

You should have something like this:

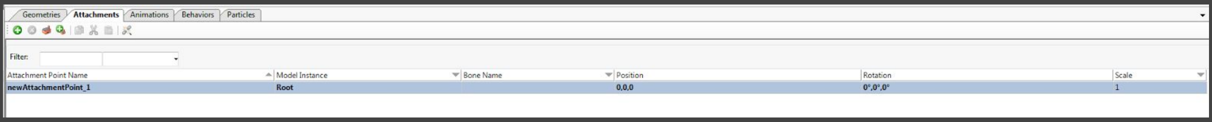


Looks nice enough, but as you can see the position of the head is kind of iffy. While you can keep this as is, we’re going to do something to fix it. Go to the Attachments tab on the Asset Editor:



Attachment Point Name	Model Instance	Bone Name
Cannon	Root	Cannon
FX_Cannon_01	Root	Cannon
FX_Fuse_01	Root	Cannon
FX_MuzzleFlash_01	Root	Cannon

As you can see, the Crouching Tiger has already a bunch of custom attachments. You can delete them since we won't be using any of them, however there's no need. Instead we're here to create a new attachment point. Click on the green Plus button to add a new attachment. You should see this:



Attachment Point Name	Model Instance	Bone Name	Position	Rotation	Scale
newAttachmentPoint_1	Root		0,0,0	0°,0°,0°	1

So with these cells we can set up our new attachment point. Please note that custom attachment points don't animate so using this for full bodies and stuff isn't going to be great. These are the bits of info we need.

- **Attachment Point Name:** Whatever name we want.
- **Model Instance:** Is usually Root, as it refers to which model it should apply into.
- **Bone Name:** You can use the button at the cell to look at all the bones of the model. In this case we need "Head"
- **Position:** There are 3 values here, corresponding to x, y and z. This allows us to slightly change the position of our attachment if we're not too happy with it.
- **Scale:** Same as above, but, you know, with rotation.
- **Scale:** Refers to the size. 1 is default.

We can now fill the cells with what we need:

- **Attachment Point Name:** Head_Fix
- **Model Instance:** Root
- **Bone Name:** Head
- **Position:** -11,2; -0,75; 0
- **Position:** 90; 0; -90
- **Scale:** 0,95

Now set the head we put earlier to this attachment instead. You should have something that looks like this:

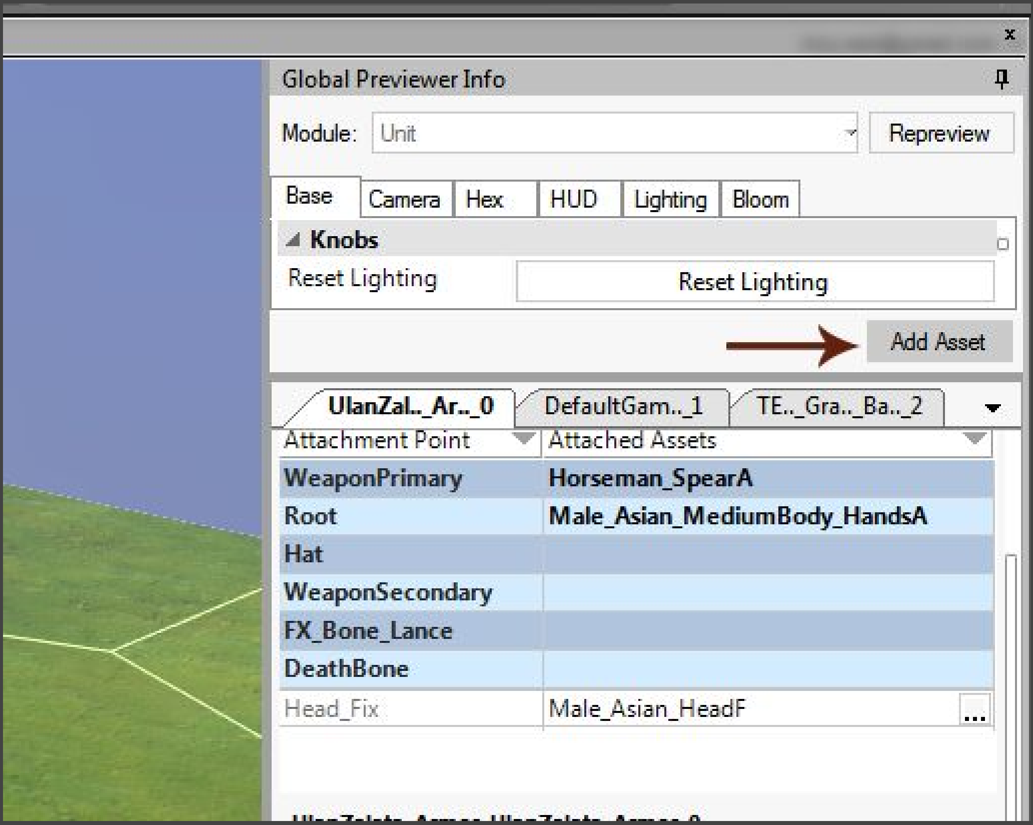
.46 KB
.26 KB
703.88 KB

So… the rider is ready, but where is the horse?

Horseback Riding 101

If you check all the attachments of our rider, there isn't a single one where we can attach the Horse. This is because the way mounted units are handled is slightly different than that of unmounted units. In the artdefine the horse and the rider are set up independently and the rider is then attached to the horse.

The problem is that this makes it a bit hard to preview. Since we have no place to put the horse at, we'd need to preview 2 different assets. In order to do this you need to find the "Add Asset" button on the Previewer. (Sometimes it hides on the UI so be sure to move everything around to find it)

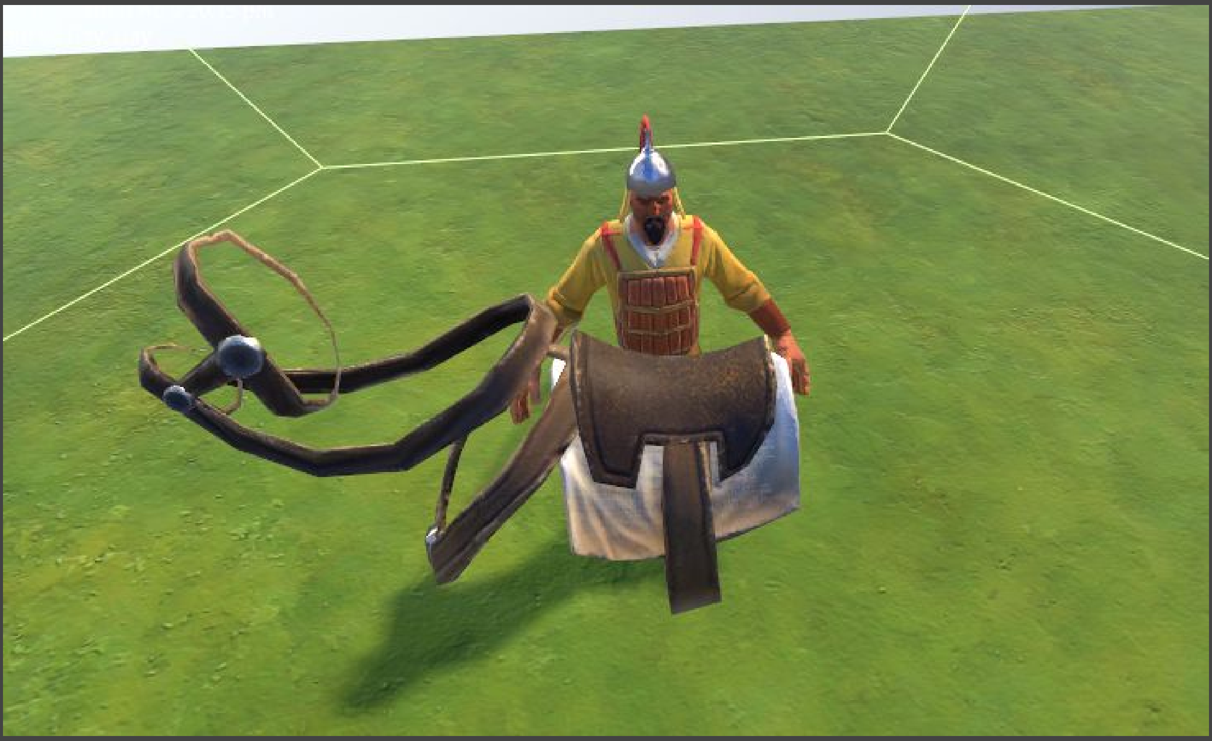

Global Previewer Info

Module: Unit

Reset Lighting	Reset Lighting

Attachment Point	Attached Assets
WeaponPrimary	Horseman_SpearA
Root	Male_Asian_MediumBody_HandsA
Hat	
WeaponSecondary	
FX_Bone_Lance	
DeathBone	
Head_Fix	Male_Asian_HeadF

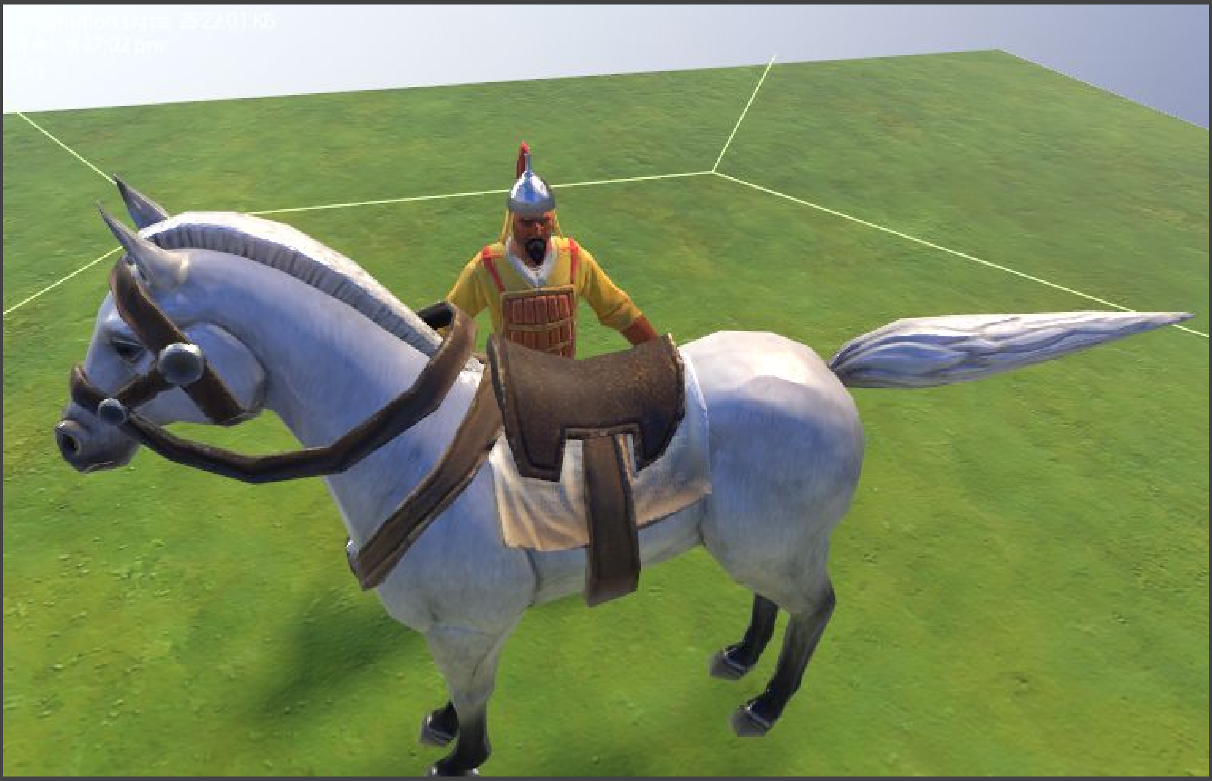
The Asset Browser box we now know so well will appear. Just like with units, the base for the asset should be the HorseArmor instead of the body. Search for Horseman_HorseArmorA and open it.



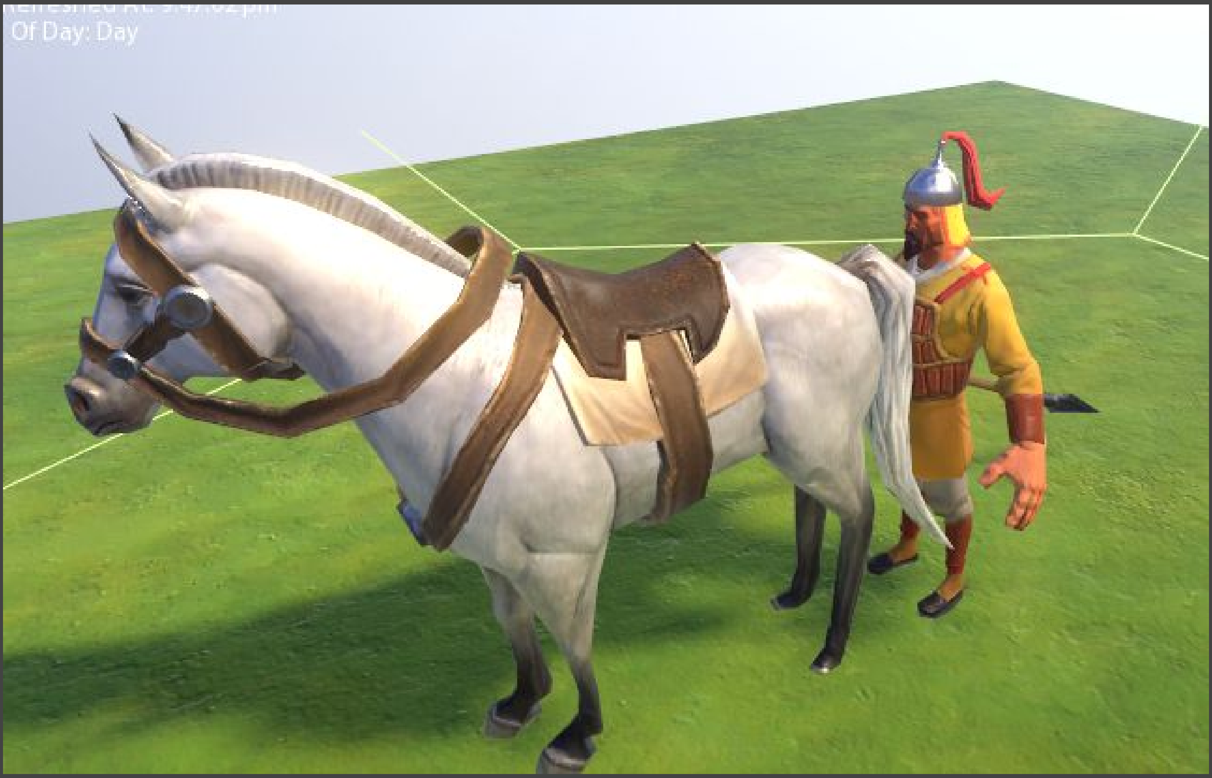
Looks a bit strange right? But that’s just how it is. On the new asset we can set up animations, attachments and tints just like before.

Now let’s add attachments to the horse armor:

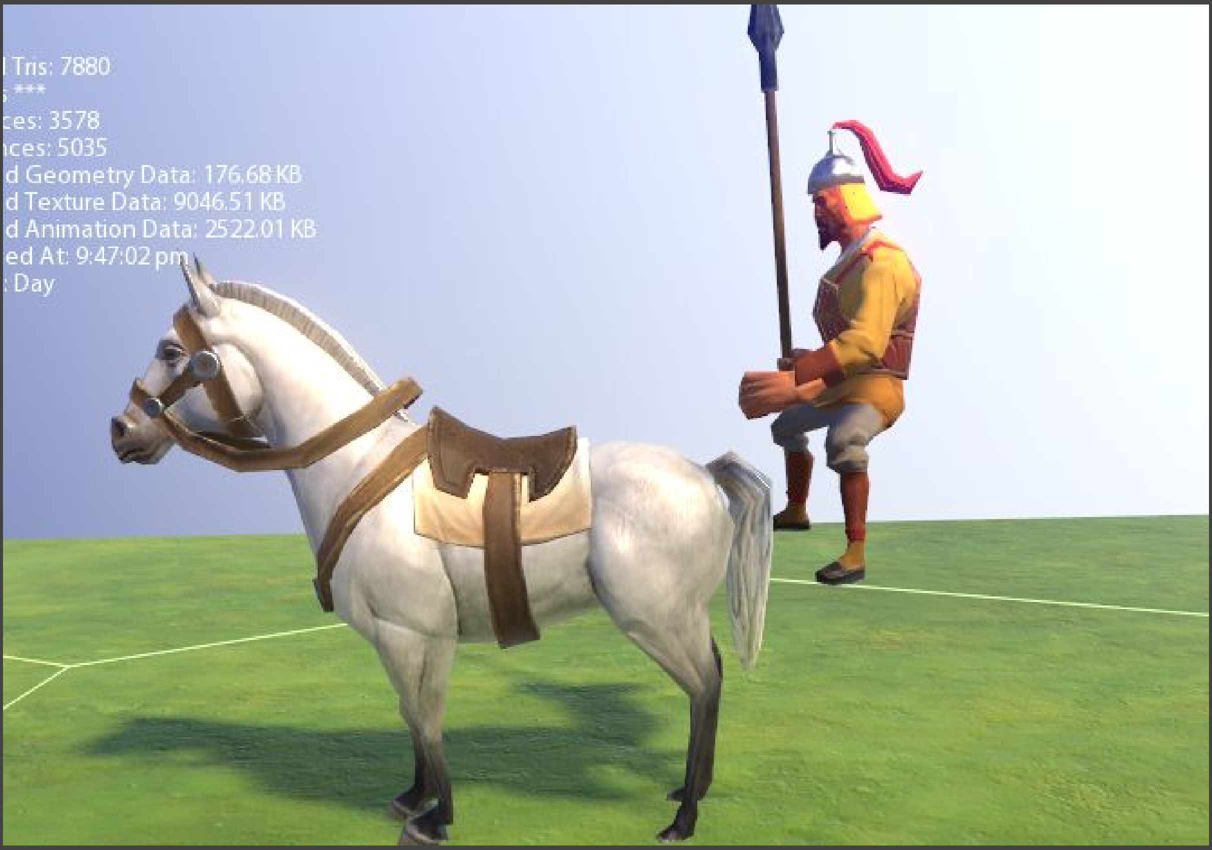
- **Root:** Horse_Medium_BodyB; Horse_Medium_Mane_C; Horse_Medium_TailA



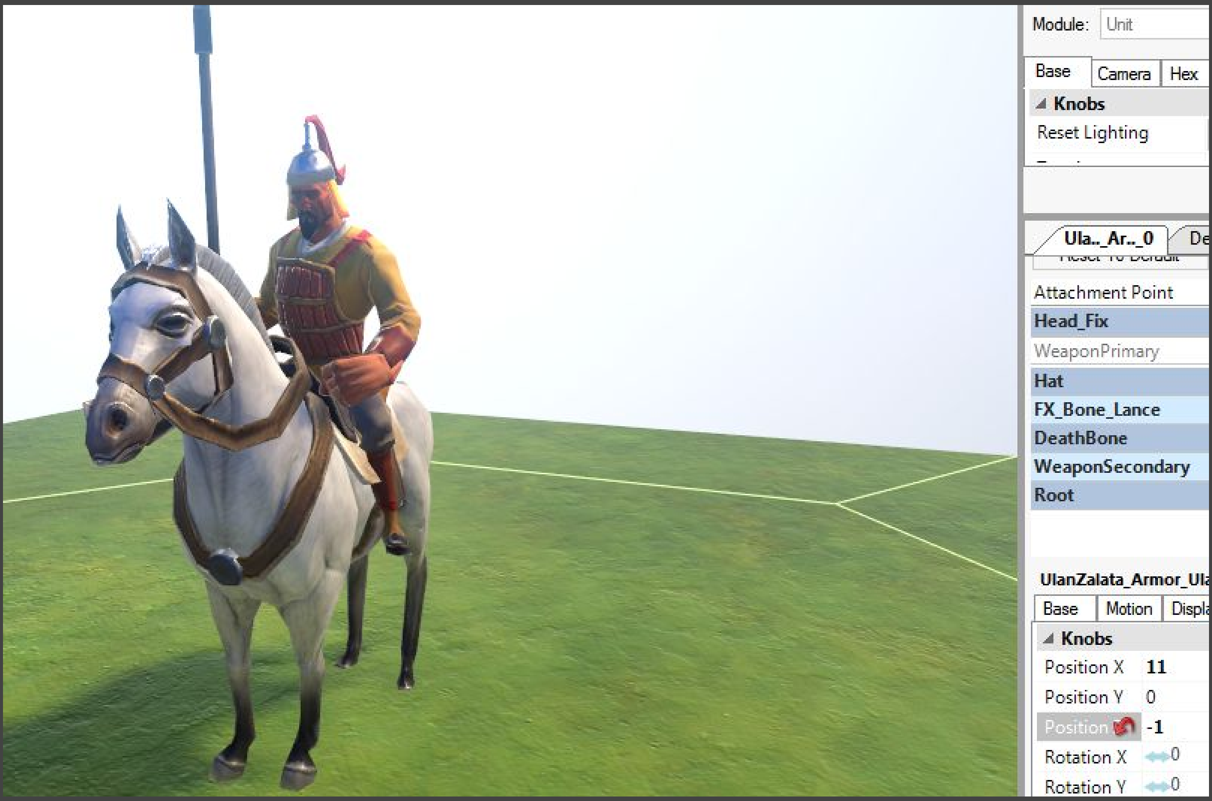
So… we now have both our Horse and our Rider ready for battle. However, life’s not perfect. Play the Idle animation of the Horse and you’ll see what I mean.



As you can see, the horse and the rider are not going to play both animations at the same time automatically. And the positioning is also off. Without stopping the Horse's idle animation, play the Riders animation.


l Tris: 7880
; ***
ces: 3578
nces: 5035
d Geometry Data: 176.68 KB
d Texture Data: 9046.51 KB
d Animation Data: 2522.01 KB
ed At: 9:47:02 pm
: Day

So yeah, the position is completely off. The way to fix this is by going to the transform tab and setting the position manually. Set the Position X to 11 and the Position Z to - 1



Position X 11
Position Y 0
Position -1
Rotation X 0
Rotation Y 0

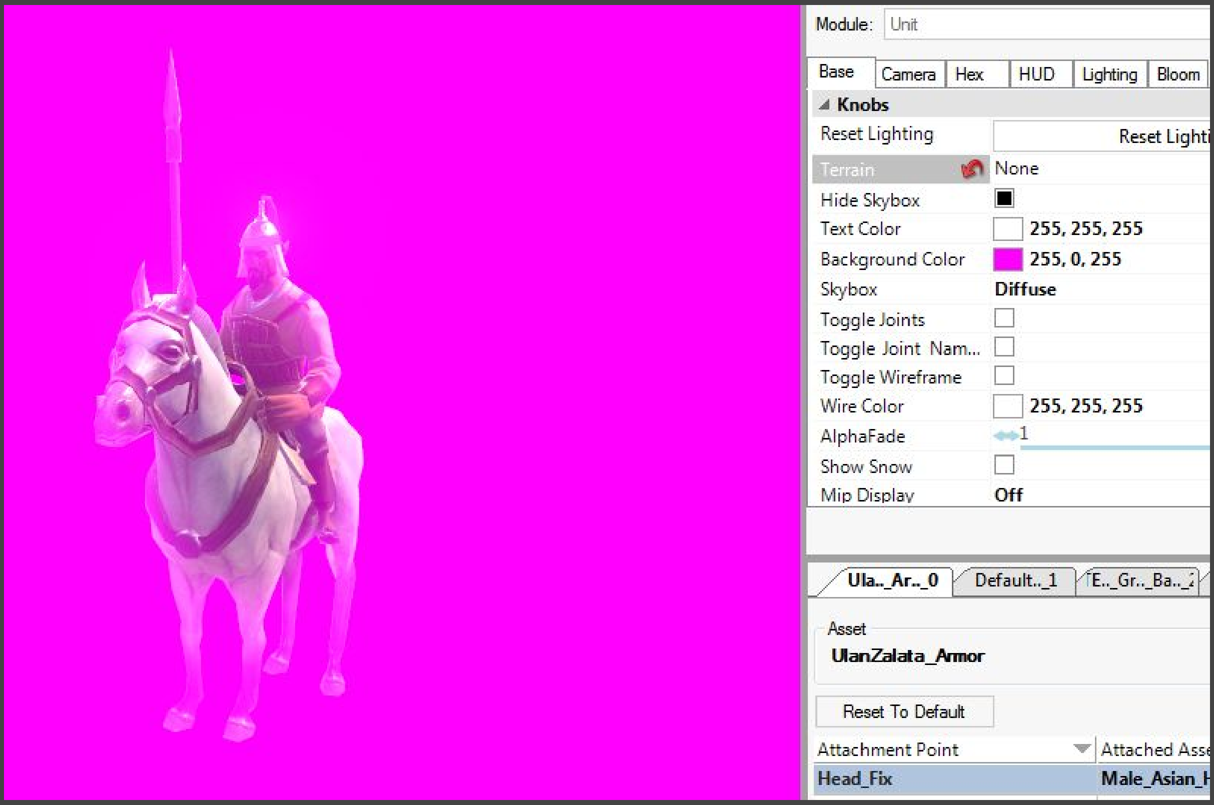
And there, we can finally see at our unit! Save your asset and rejoice!

Preparing an icon

Using the Previewer

It is during this step in which it is the most adequate idea to take a screenshot for the icon. We need to do some setup on the Previewer before that.

On the top right side of the Previewer you should see the Base tab. There we need to set the following options:

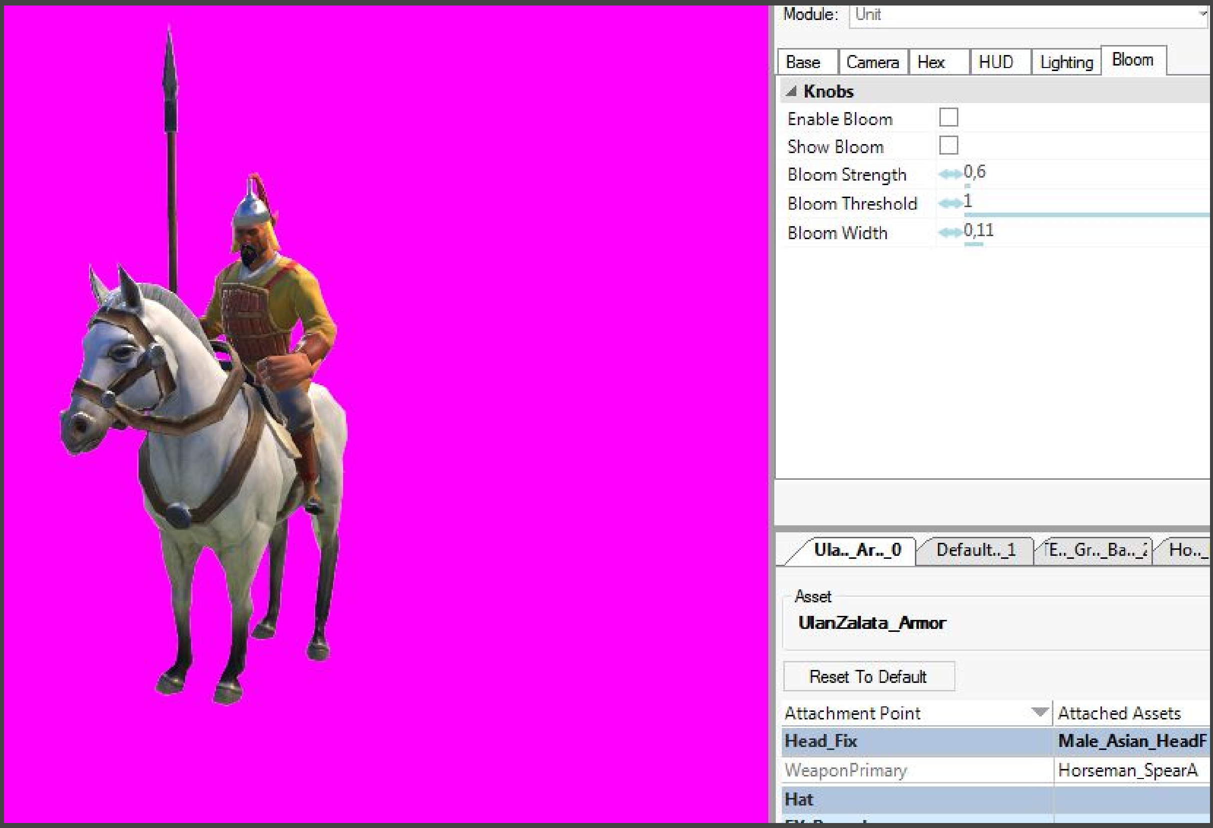


Reset Lighting	
Terrain	None
Hide Skybox	■
Text Color	255, 255, 255
Background Color	255, 0, 255
Skybox	Diffuse
Toggle Joints	☐
Toggle Joint Nam...	☐
Toggle Wireframe	☐
Wire Color	255, 255, 255
AlphaFade	1
Show Snow	☐
Mip Display	Off

Asset
UlanZalata_Armor

Attachment Point	Attached Asse
Head_Fix	Male_Asian_H

Next go to the Bloom tab and disable it.



Module: Unit

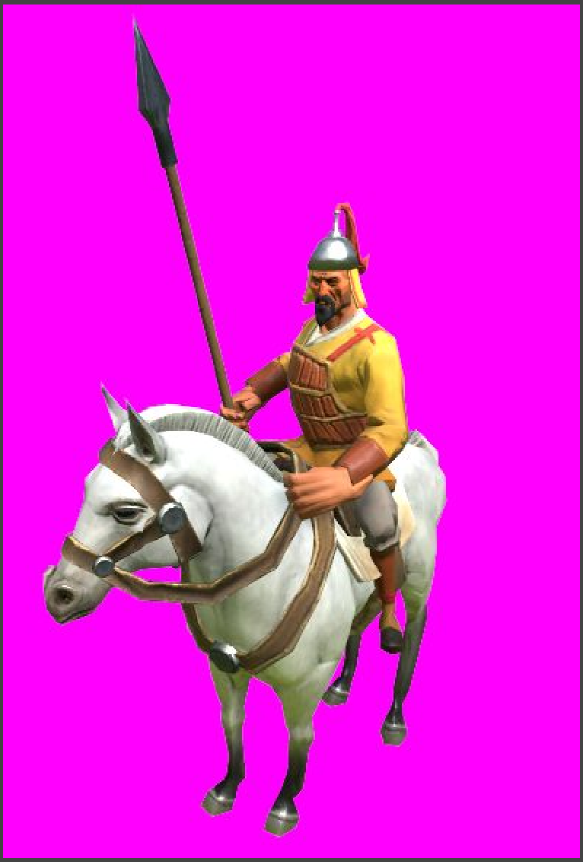
Knobs

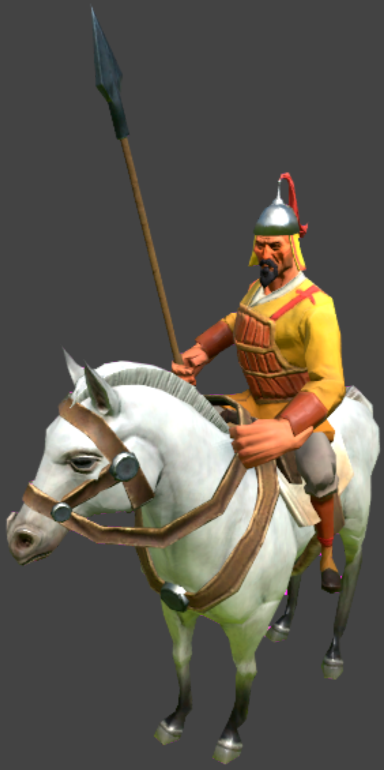
Enable Bloom	☐
Show Bloom	☐
Bloom Strength	0,6
Bloom Threshold	1
Bloom Width	0,11

Asset
UlanZalata_Armor

Attachment Point	Attached Assets
Head_Fix	Male_Asian_HeadF
WeaponPrimary	Horseman_SpearA
Hat	

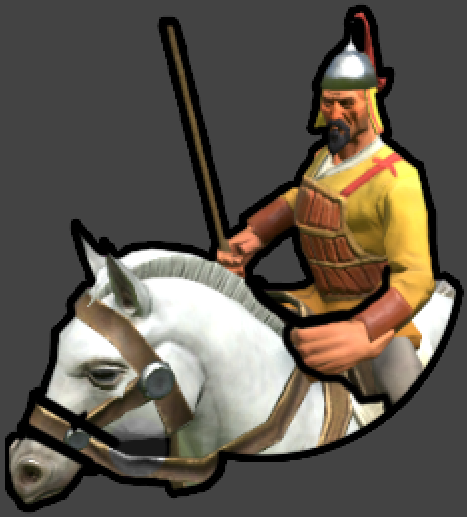
Almost done. Now we have to change the lighting to one that doesn’t obscure the face. Please note that when changing the game environment all animations will be reset (sadly). You can use whatever light you see fit, but personally I find “StadiumCenter_Game” good for icon making. Remember to play your animation and set your position again and you should have this:





So… now that we have our screenshot we can use Photoshop or Gimp to remove the background. And then we can use a template such as this to make the icons.

The result should look like this:



You need to save icons in TGA or DDS format keeping the transparency as alpha channels. (I personally recomend TGA to prevent quality loss) The sizes required for units are the following: 256, 200, 95, 70, 50, 38.

For the sake of continuing with the tutorial download the following files with the icons and flags in all required sizes:

SUPER IMPORTANT FILE

XLPs

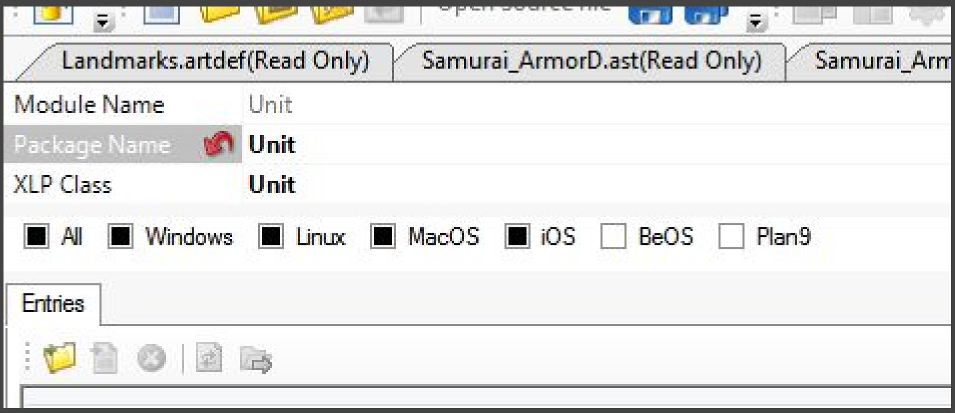
Introduction to XLPs

So, we have a custom asset, and a bunch of icons. But how do we make them be recognized by the game? The answer is XLPs.

Whenever you have custom assets and textures you need to make the XLPs before proceeding with the artdefines. XLPs are basically package files that will be cooked by the game when delivering the final mod.

In order to make a XLP you must go to the asset editor and choose New from the File menu. The option to make an XLP will appear (just like when making artdefines)

XLPs come in different classes depending on your needs. First we'll create a XLP for the new asset we created. On the xlp you must fill this information:



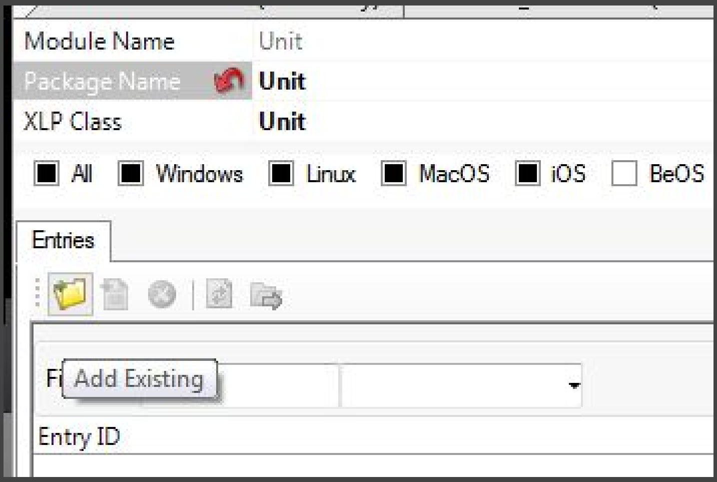
Module Name	Unit
Package Name	**Unit**
XLP Class	**Unit**

■ All ■ Windows ■ Linux ■ MacOS ■ iOS □ BeOS □ Plan9

One can change the Package Name to anything, but I prefer using the XLP class name as the Package Name as it seems to me it prevents crashes and bugs. (might be a placebo though)

Before doing anything else, save the XLP in the XLP folder of your project. **YOU MUST SAVE IT WITH THE SAME NAME AS THE PACKAGE NAME PLACED ABOVE**, otherwise errors will appear and you'll be dead.

Once it finishes saving you can add our new asset by pressing the "Add Existing" button.



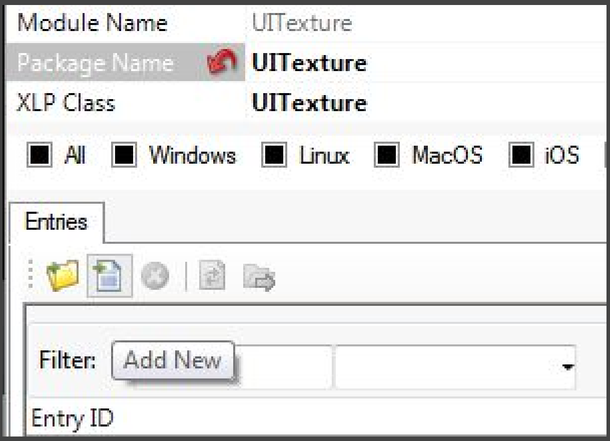
Module Name: Unit
Package Name: Unit
XLP Class: Unit

■ All ■ Windows ■ Linux ■ MacOS ■ iOS ☐ BeOS

The Asset Browser will appear, and you should be able to find our UlanZalata_Armor asset. Add it and save the XLP again.

Now we'll do the same for our icons (which I hope you remembered to make/download)

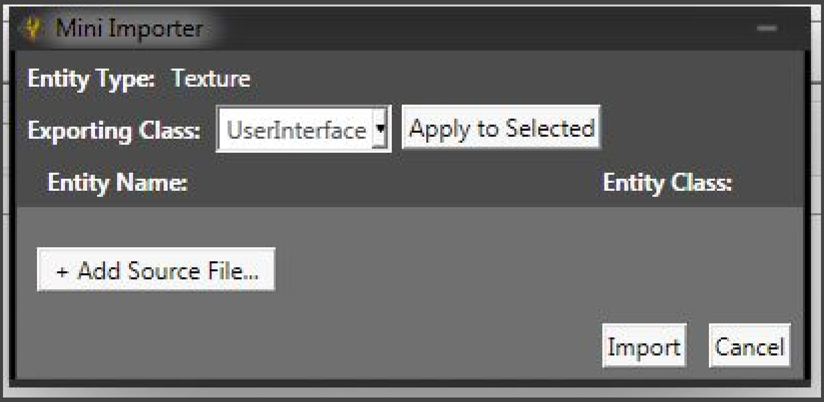
Create a new XLP of the UITexture class and save. This time instead of the "Add Existing" button we'll need to use the "Add New" that is right beside it.



Module Name: UITexture
Package Name: UITexture
XLP Class: UITexture

■ All ■ Windows ■ Linux ■ MacOS ■ iOS

The Mini Importer popup should appear:



There you can click on +Add Source File to put any DDS or TGA file you want to add to your mod. Something very important to know is that **ONCE A FILE IS ON THE XLP, IT IS FOREVER LINKED TO ITS SOURCE FILE.** This means that if you change the source file (say, you decided to change a little detail on an icon and you replace the file directly), a bunch of errors will appear because the XLP is not up to date with the source file. In order to modify your Source Files you should first remove them from your project and from the XLP.

Anyway, add all the icons we require and press Import. Then save your XLP file.

Some more info

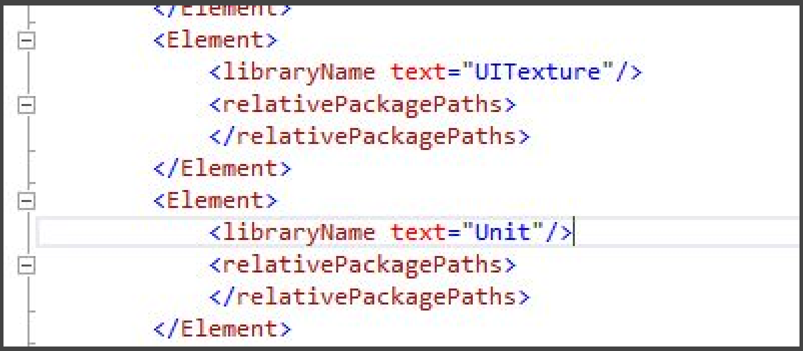
When you import files to the XLP, your project will now have all of them in a Textures Folder (check in explorer if you do not see them on modbuddy). What importing does is convert any TGA to DDS, and then it creates a .tex file which is how the game interprets the .dds

Both files have the same name, and must not be touched unless you need to modify the textures (in which case, again, I recommend deleting them and their reference on the xlp and reimport).

Modart File shenanigans

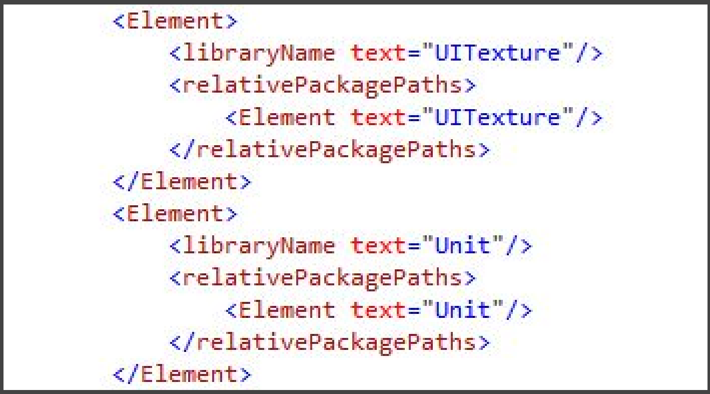
Just like artdefs, it is very important to have xlps referenced on the mod.art.xml file.

They go on a different section below the artdefs. You should see a place such as this:



```
        </Element>
        <Element>
            <libraryName text="UITexture"/>
            <relativePackagePaths>
            </relativePackagePaths>
        </Element>
        <Element>
            <libraryName text="Unit"/>
            <relativePackagePaths>
            </relativePackagePaths>
        </Element>
```

Between the relativePackagePaths we need to put the names of our xlps like this:



```
<Element>
    <libraryName text="UITexture"/>
    <relativePackagePaths>
        <Element text="UITexture"/>
    </relativePackagePaths>
</Element>
<Element>
    <libraryName text="Unit"/>
    <relativePackagePaths>
        <Element text="Unit"/>
    </relativePackagePaths>
</Element>
```

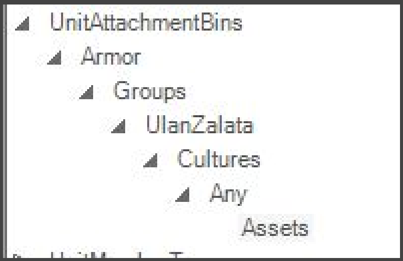
Save the mod.art.xml file and we may proceed with the ArtDefines

Mounted Art Defines

UnitBins with custom assets

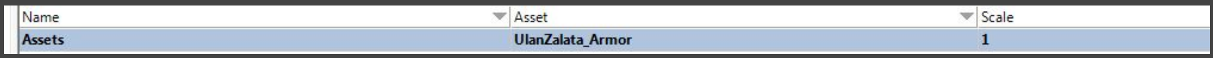
So, we already know how to make Art Defines, but in the case of mounted units there is a slight difference.

Create a new artdef file called Units.Artdef. We'll first need to make a unit bin with our new custom asset. Use the following structure:



```
◢ UnitAttachmentBins
    ◢ Armor
        ◢ Groups
            ◢ UlanZalata
                ◢ Cultures
                    ◢ Any
                        Assets
```

On the Assets category you'll find 3 columns (Name, Asset, Scale)... Fill the Asset scale with our custom asset's name (UlanZalata_Armor). You must type the name manually because the BLP entry browser sometimes doesn't load custom assets. If you made the xlp file right, you should be able to type the name of the Asset and then it will stay like that.



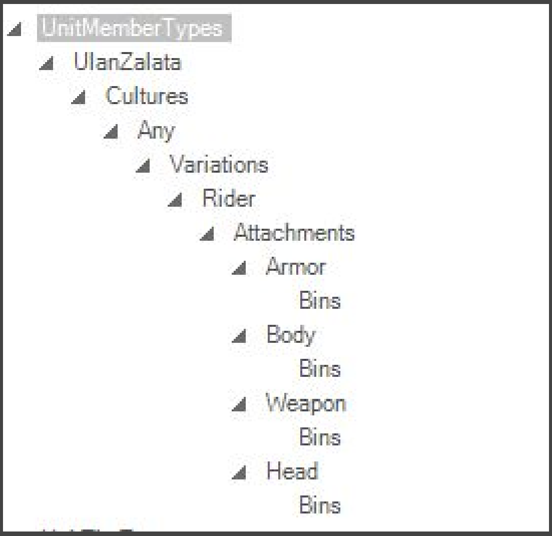
Name	Asset	Scale
Assets	UlanZalata_Armor	1

Now that we have our new UnitBin, let's set up our UnitMember.

Setting the Rider

So… remember how we set up a Custom Attachment on the asset for our head fix? This is the part where we actually get to use it.

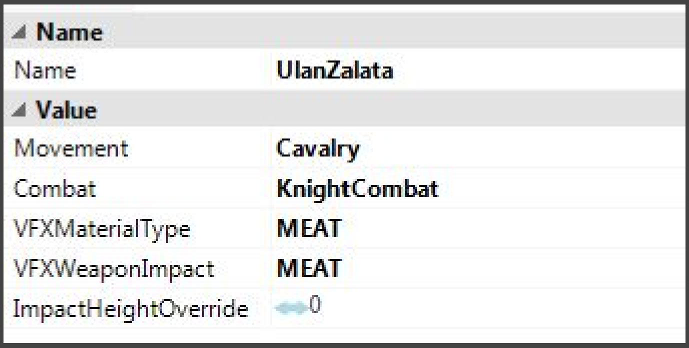
Create a new UnitMemberType with the following hierarchy:



```
UnitMemberTypes
  UlanZalata
    Cultures
      Any
        Variations
          Rider
            Attachments
              Armor
                Bins
              Body
                Bins
              Weapon
                Bins
              Head
                Bins
```

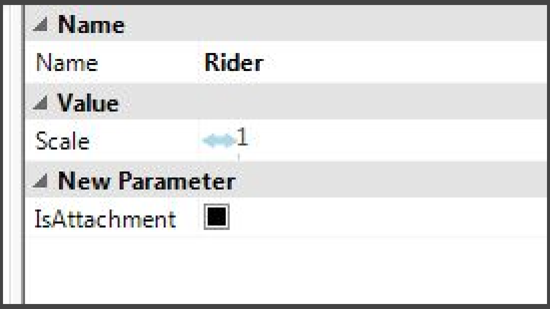
Something very important to note here is that the name of the variation is "Rider", this is how we'll recognise it when we setup the horse.

Under UlanZalata you need this information (taken from the Horseman unit):



Name	
Name	**UlanZalata**
Value	
Movement	**Cavalry**
Combat	**KnightCombat**
VFXMaterialType	**MEAT**
VFXWeaponImpact	**MEAT**
ImpactHeightOverride	0

Then under Rider, you need this:



Name	
Name	**Rider**
Value	
Scale	1
New Parameter	
IsAttachment	■

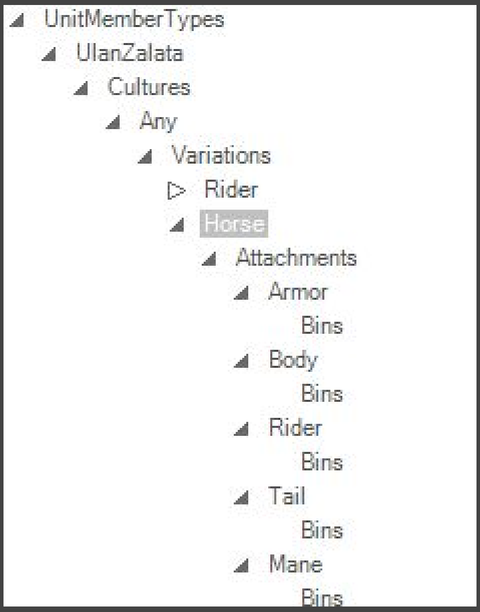
Checking the IsAttachment box is very important for mounted units

Finally, these is the info you need to put on each attachment:

- **Armor:**
  - **Point:** Root
  - **Bins:** Armor/UlanZalata (the one we added a bit ago)
- **Body**
  - **Point:** Root
  - **Bins:** Bodies/Male_MediumBody_HandsB
- **Weapon**
  - **Point:** WeaponPrimary
  - **Bins:** Spears/Horseman
- **Head**
  - **Point:** Head_Fix (the custom attachment we made)
  - **Bins:** Heads/Male

Setting the Horse

So, we now have the Rider set up. In order to add the horse we need to add it as a Variation.



```
◢ UnitMemberTypes
   ◢ UlanZalata
      ◢ Cultures
         ◢ Any
            ◢ Variations
               ▷ Rider
               ◢ Horse
                  ◢ Attachments
                     ◢ Armor
                          Bins
                     ◢ Body
                          Bins
                     ◢ Rider
                          Bins
                     ◢ Tail
                          Bins
                     ◢ Mane
                          Bins
```

And this is how the Attachments are set up:

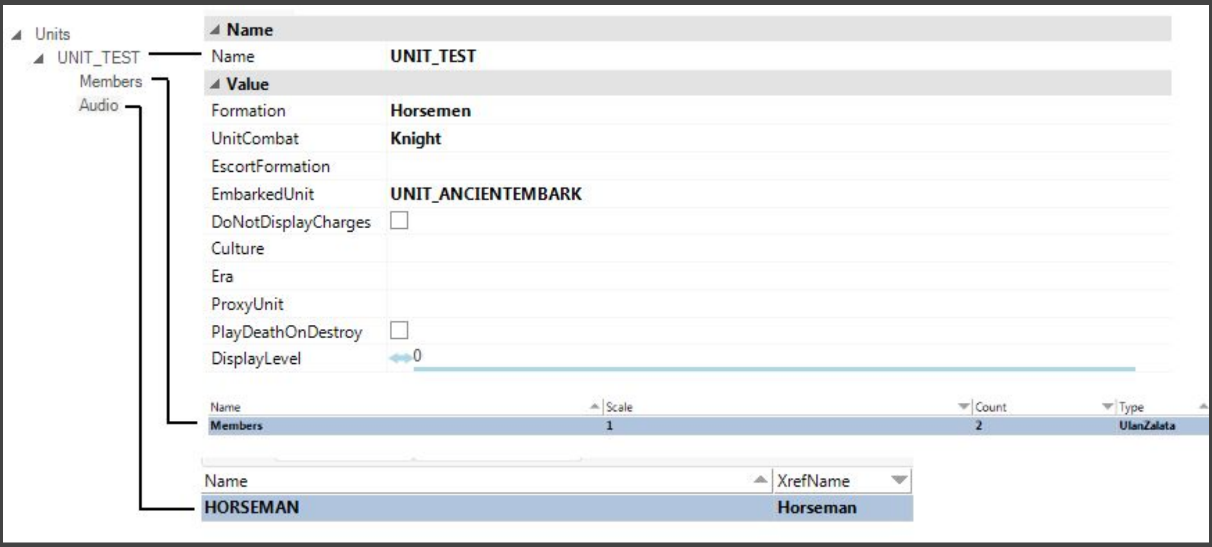
- **Armor**
  - **Point:** Root
  - **Bins:** Armor_Horse/Horseman
- **Body**
  - **Point:** Root
  - **Bins:** Animals/Horse_Medium
- **Tail**
  - **Point:** Root

    - **Bins:** Accessories_Horse/Tails
- **Mane**
    - **Point:** Root
    - **Bins:** Accessories_Horse/Manes
- **Rider**
    - **Point:** RiderAttach
    - **Bins:** V:Rider *(THIS is how the Rider is referenced, V: … allows for a Variation set as an attachment to be placed on the armor)*

And with that our UnitMember is done :D

Unit

So time to set up the Unit artdef. This time there's nothing new compared to the last level, so this is pretty much the hierarchy:



- Units
    - UNIT_TEST
        - Members
        - Audio

Name

Name	**UNIT_TEST**

Value

Formation	**Horsemen**
UnitCombat	**Knight**
EscortFormation	
EmbarkedUnit	**UNIT_ANCIENTEMBARK**
DoNotDisplayCharges	☐
Culture	
Era	
ProxyUnit	
PlayDeathOnDestroy	☐
DisplayLevel	0

Name	Scale	Count	Type
Members	**1**	**2**	**UlanZalata**

Name	XrefName
HORSEMAN	**Horseman**

Remember that mounted units only show 2 members at the same time!

Now save your artdefs. You may now close the Asset Editor. We're almost ready!

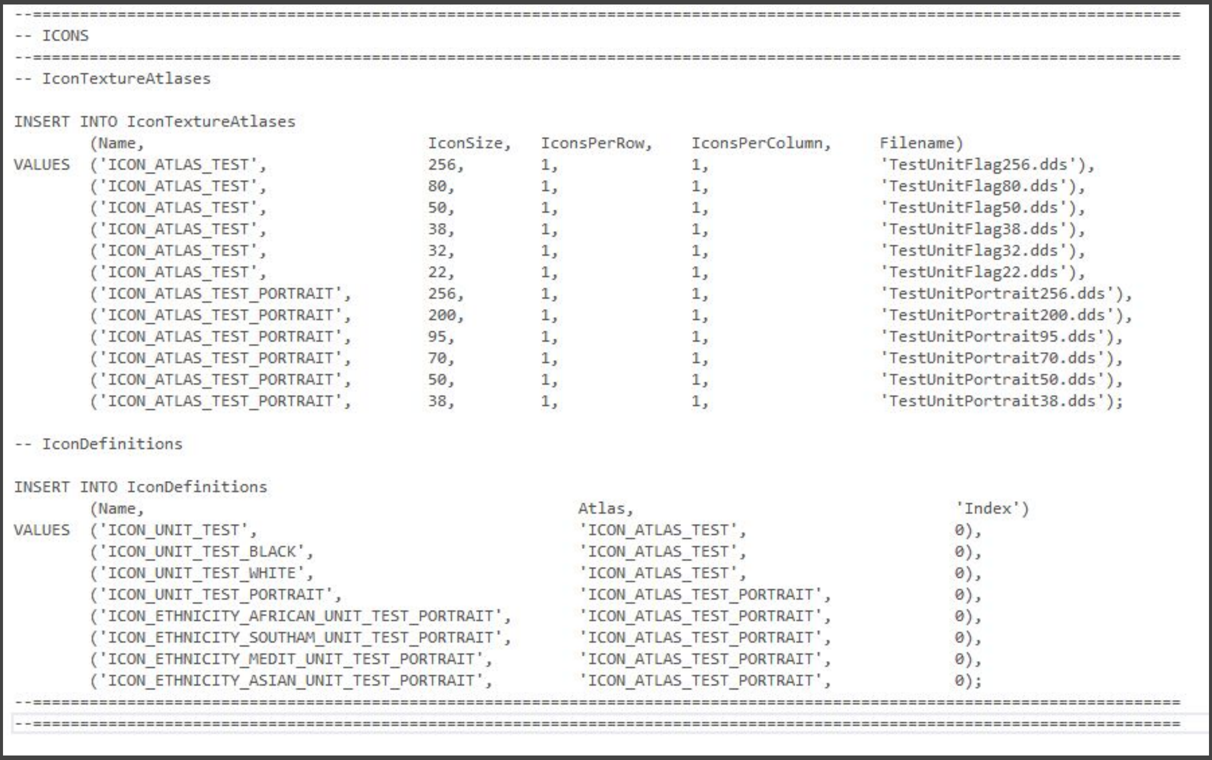
Icon SQL

Adding Icons to the Database

So, we now have our artdefs and xlps ready. But in order for our custom icons to be recognized by our unit we need to set them up on a xml or sql file (And everyone knows SQL is far superior).

On ModBuddy you'll see a folder called sql. Add a new sql file and call it TestUnitIcons.sql

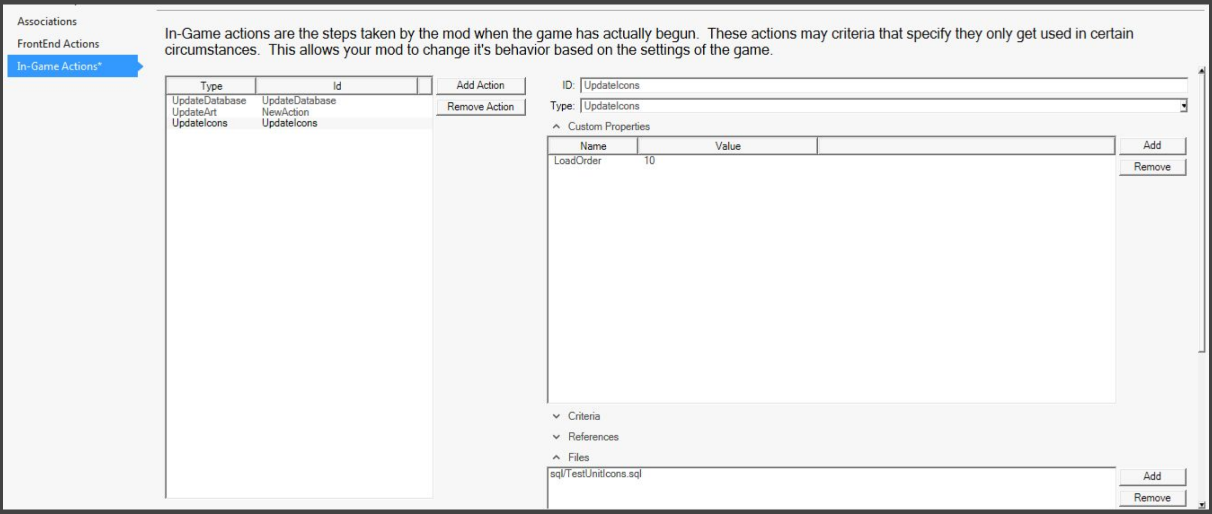
This is the content you need:



```
--==========================================================================================================================
-- ICONS
--==========================================================================================================================
-- IconTextureAtlases

INSERT INTO IconTextureAtlases
        (Name,                          IconSize,   IconsPerRow,    IconsPerColumn,     Filename)
VALUES  ('ICON_ATLAS_TEST',             256,        1,              1,                  'TestUnitFlag256.dds'),
        ('ICON_ATLAS_TEST',             80,         1,              1,                  'TestUnitFlag80.dds'),
        ('ICON_ATLAS_TEST',             50,         1,              1,                  'TestUnitFlag50.dds'),
        ('ICON_ATLAS_TEST',             38,         1,              1,                  'TestUnitFlag38.dds'),
        ('ICON_ATLAS_TEST',             32,         1,              1,                  'TestUnitFlag32.dds'),
        ('ICON_ATLAS_TEST',             22,         1,              1,                  'TestUnitFlag22.dds'),
        ('ICON_ATLAS_TEST_PORTRAIT',    256,        1,              1,                  'TestUnitPortrait256.dds'),
        ('ICON_ATLAS_TEST_PORTRAIT',    200,        1,              1,                  'TestUnitPortrait200.dds'),
        ('ICON_ATLAS_TEST_PORTRAIT',    95,         1,              1,                  'TestUnitPortrait95.dds'),
        ('ICON_ATLAS_TEST_PORTRAIT',    70,         1,              1,                  'TestUnitPortrait70.dds'),
        ('ICON_ATLAS_TEST_PORTRAIT',    50,         1,              1,                  'TestUnitPortrait50.dds'),
        ('ICON_ATLAS_TEST_PORTRAIT',    38,         1,              1,                  'TestUnitPortrait38.dds');

-- IconDefinitions

INSERT INTO IconDefinitions
        (Name,                                          Atlas,                          'Index')
VALUES  ('ICON_UNIT_TEST',                              'ICON_ATLAS_TEST',              0),
        ('ICON_UNIT_TEST_BLACK',                        'ICON_ATLAS_TEST',              0),
        ('ICON_UNIT_TEST_WHITE',                        'ICON_ATLAS_TEST',              0),
        ('ICON_UNIT_TEST_PORTRAIT',                     'ICON_ATLAS_TEST_PORTRAIT',     0),
        ('ICON_ETHNICITY_AFRICAN_UNIT_TEST_PORTRAIT',   'ICON_ATLAS_TEST_PORTRAIT',     0),
        ('ICON_ETHNICITY_SOUTHAM_UNIT_TEST_PORTRAIT',   'ICON_ATLAS_TEST_PORTRAIT',     0),
        ('ICON_ETHNICITY_MEDIT_UNIT_TEST_PORTRAIT',     'ICON_ATLAS_TEST_PORTRAIT',     0),
        ('ICON_ETHNICITY_ASIAN_UNIT_TEST_PORTRAIT',     'ICON_ATLAS_TEST_PORTRAIT',     0);
--==========================================================================================================================
--==========================================================================================================================
```

So as you can see, one can set different ethnicities for icons. For now we can just assign the icon we made to all ethnicities. The game will recognize these as the icons by the TEST (everything that is after UNIT_) automatically.

Go to the mod properties and add the UpdateIcons In-Game Action



Associations
FrontEnd Actions
In-Game Actions*

In-Game actions are the steps taken by the mod when the game has actually begun. These actions may criteria that specify they only get used in certain circumstances. This allows your mod to change it's behavior based on the settings of the game.

Type	Id
UpdateDatabase	UpdateDatabase
UpdateArt	NewAction
UpdateIcons	UpdateIcons

Add Action
Remove Action

ID: UpdateIcons
Type: UpdateIcons

^ Custom Properties

Name	Value
LoadOrder	10

Add
Remove

v Criteria
v References
^ Files

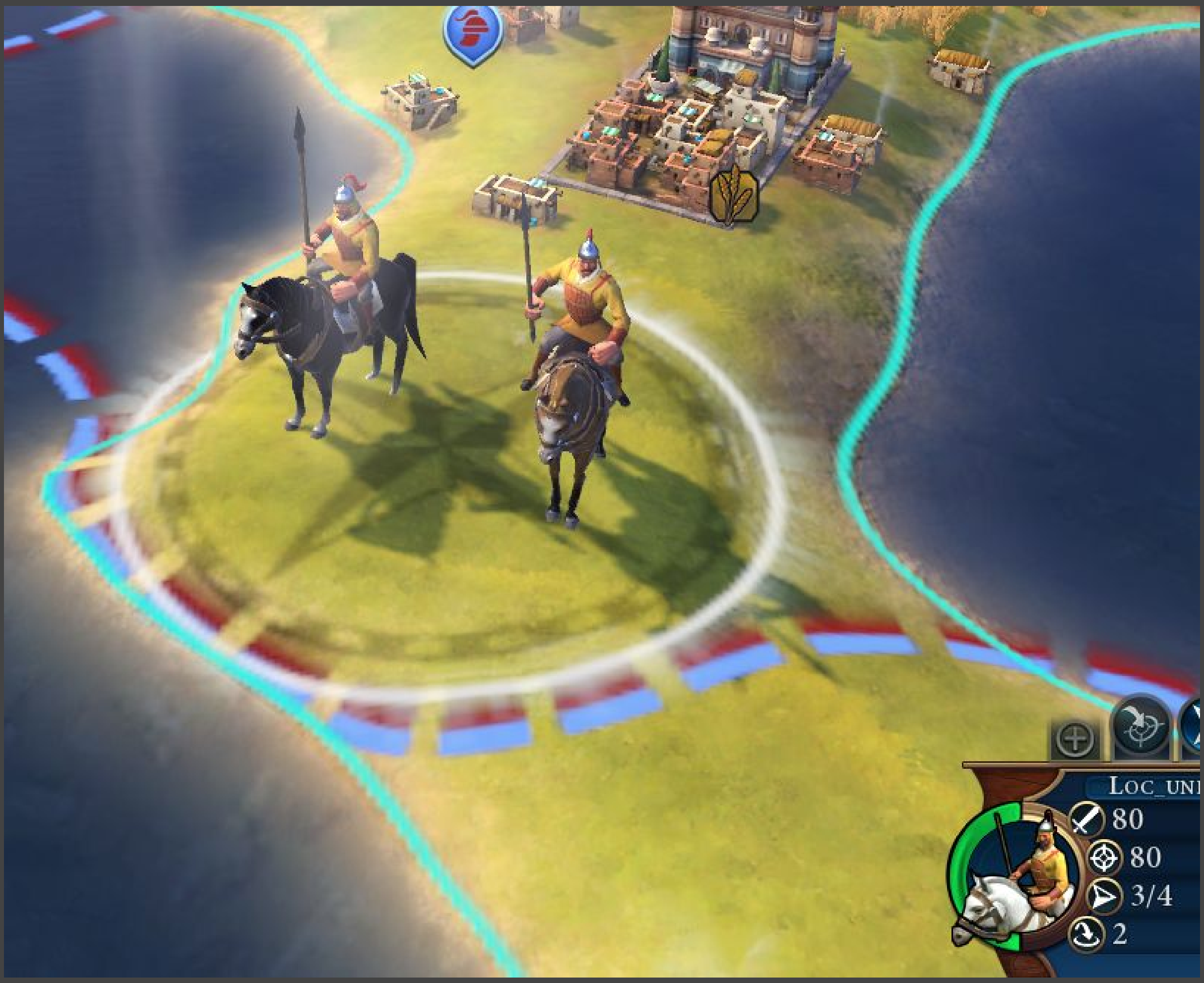
sql/TestUnitIcons.sql

Add
Remove

Before building the mod, remember to check if all references to the XLPs and Artdefs are in the mod.art.xml (hint: the Units.artdef might not be in)

Build the mod and now we're ready to test!

Rejoice



If you did everything fine, you should have our unit, icon and flag all displaying on the game. Rejoice for **you have passed Level 2!** Congratulations as you can now make virtually every combination possible that don't require custom textures or models! Yay!