Getting your units in Civilization 4:
A guide for anyone with Max and determination

What follows is the methodology to go from model to animated unit in Civilization 4.

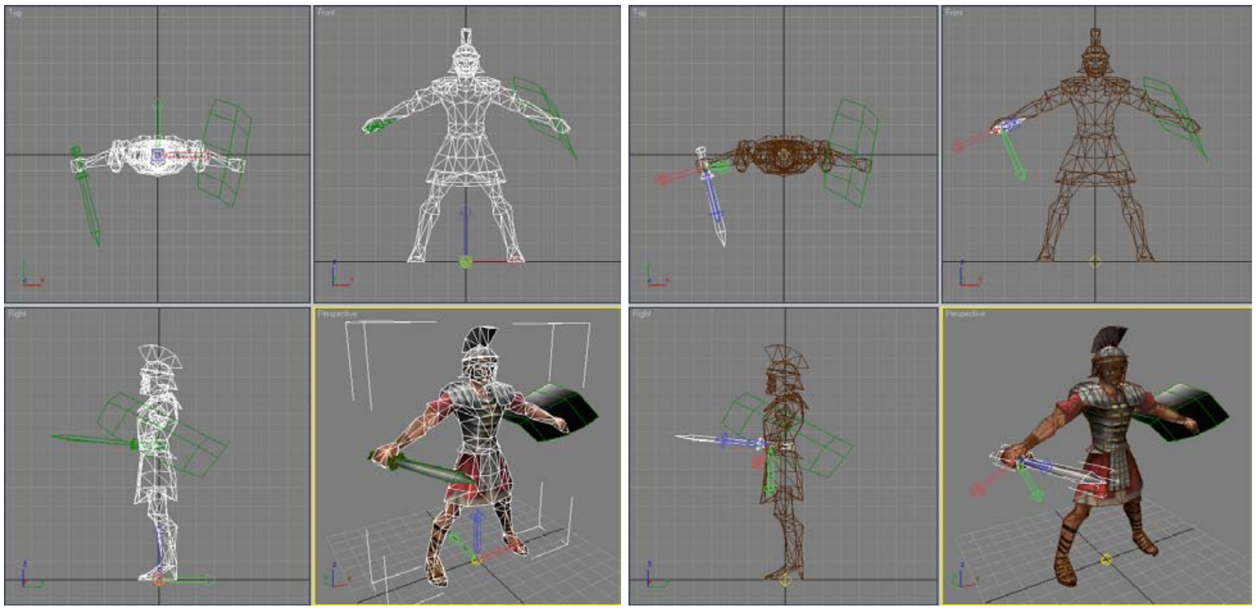
Make a Model

First things first, begin with a model that rocks. Follow this checklist for good practices when modeling for real time stuff.

- Try to keep your model at or under 1000 verts.
- Have as few unique objects as possible, and do not use multi-sub object materials.
- Use .dds textures for your maps, and try not to use more than 3 unique textures per model.
- Use Roman_Praetorian_Model.max as an example of how to do a clean unwrap, and use team color that shows up in the game. This model was made by Brian Busatti. Thank you Brian for making models that are fun to animate.

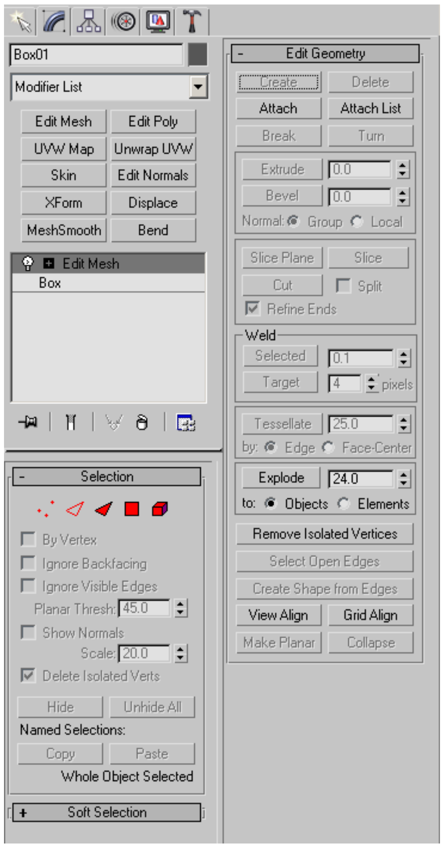
Get Model Ready for Prime Time

There are a few things you should do to that model to make sure the skinning process works the way you expect it to.



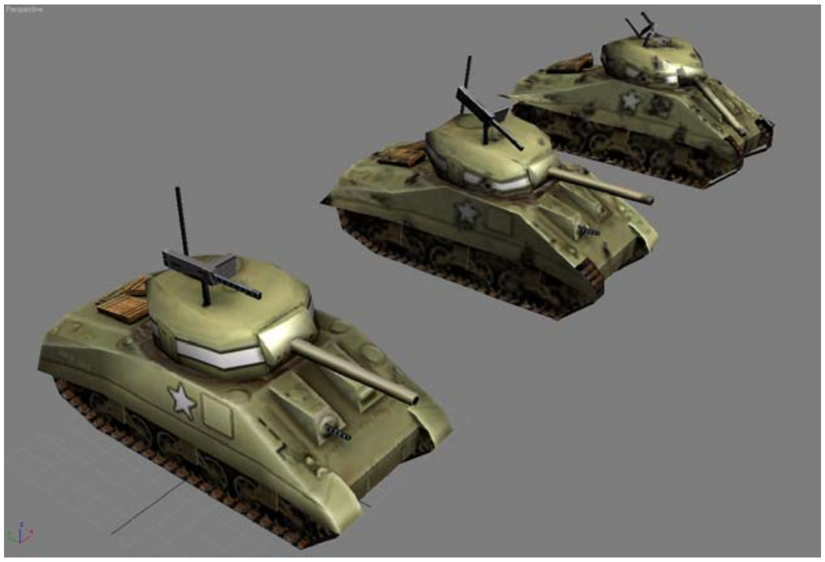
Above you will see screen grabs of the Roman Praetorian. In the image to the left you will see that the model of the soldier is centered in the X and Y space with its feet placed on the 0 of the Z height. The actual pivot point of the soldier is exactly at 0, 0, 0 in world space. The sword is handled slightly differently with the pivot point of the sword aligned to the axis of the sword, and is placed near the handle (this placement makes animating the sword a very intuitive process).

One thing you will want to do is make sure your model is nice and clean. Because of the way 3D data is managed I 3DS Max artist working on models for animation reset you object’s transform data; resetting all values of position, rotation, and scale. I recommend doing the following for each object in your scene. Create a box and align it to where you want your pivot point to be for the object. Apply the object’s material to this new box. Select this box, and add an edit mesh modifier to it.



Now click on the attach button, and select the object you are cleaning up. Select the modify element button, delete the geometry of the box, and name your object something meaningful. Move on and do the following to each object. Now your awesome Civilization 4 model is ready to begin the rigging and skinning process.

If you are making a tank, a boat, an airplane, or whatnot take a look at the Tank_Model.max made by our illustrative artist Rob Cloutier. You will notice the textures are handled slightly differently to handle the damage states, which are not needed for the character units. This is a good time to make damage states of your model.



Take a look at Tank_Model_and_Damage_State_Models.max and you will see the base tank model and two damage states based on that model. In Civilization 4 each mechanical unit has 3 models to represent 4 damage states: Damage state one is the default unit, damage state two is the slightly damaged unit, damage state three is the totally damaged model, and damaged state four is damage state three with smoke emitting from the rear.

To make these damage states you will want to copy your base model, and move the verts and faces of the model around to to achieve bends and dents. You will then also make a damage texture which will overlay the base texture where you will add smoke damage, bullet holes, and any other destruction you wish to add.

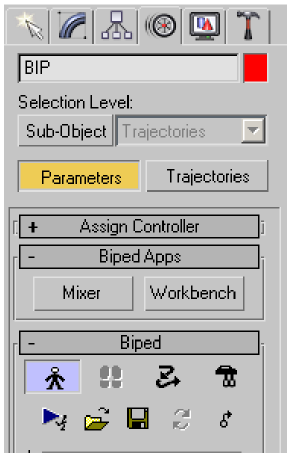
For the most part, you can achieve a really good look without needing to modify the model at all. If you are looking to save a little time, instead of making model changes, focus primarily on good damage looks to your textures. We'll take a look at how to handle damage states in the Rigging Section for Mechanical units.

Make a Rig for Character Units

If you plan on animating a human I recommend using this character file as the source for your biped: Roman_Praetorian_Biped_Rig.max. Open this file up and merge your model into the scene. This file will make getting up and running much easier. There are a full set of animations included that should work in Civilization 4 on export. There are also links that allow you to add sounds as well as attaching shared effects to your units like footsteps and explosions. I'll highlight these as needed, but it's just to underscore the advantage of starting with our stuff instead of from scratch.

You'll want you model's scale to be be pretty close to the biped's. If they are off by a lot, select your models individually, go into their sub-object vertex mode, select all the verts, and scale accordingly. Doing it this way won't muck up all the good modelling procedures you've followed thus far. After they look pretty close you'll be done with the model until it's time to skin it.

The next procedure will be to closely align the Biped with your model so that the joints bend in the right locations. You'll want to make sure the biped is in Figure mode. This mode is the default mode that stores the starting rotation and position states of each limb in the character. You'll know you are in figure mode if the button with the stick figure in it is blue. Take a look at Roman_Praetorian_Rig_Alinment.max to see where we decided to put the joints in our models. This should be a good standard for you to follow to get predictable results.



While the Biped is in Figure mode you will want to first position the BIP object so that the hips and leg joints are in the right place. Now work you way down the legs to the feet. You'll want to use a combination of non-uniform scale and rotate to make sure everything lines up 100%. If your model is symetrical, which is usually a good idea, you can position just one side of the biped, and mirror paste to get both legs in the right place. Now move up to the spine and head, and finish by getting the arms aligned. Make sure you rotate around because it's fairly common to think everything is in the right place, and then realize that from the side nothing is quite right.

Make a Rig for Mechanical Units

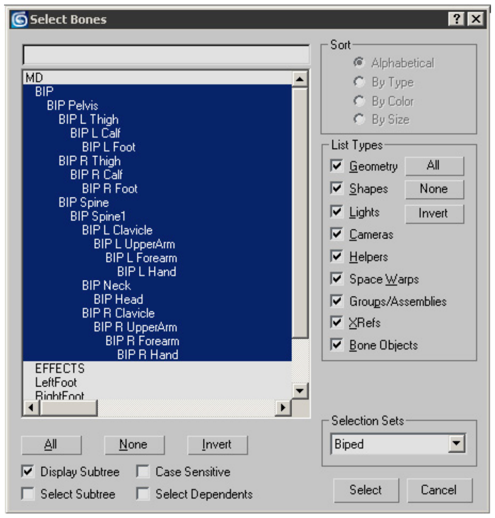
Beginning a Rig for a mechanical object is a little trickier, because there aren't ready made biped like rigs and there is not a figure mode for alignment purposes. Open the file Tank_Rig_For Simple Damage States.max and take a look at the structure. There are a bunch of Dummy objects that are hidden which handle sound effects, explosions and smoke. You can unhide these after looking at the basic rig structure.

The Tank body is given its own box to for its postion and rotation. The turret is linked to this, and from the turret are two objects, one which controls the angle of the cannon and another which controls the angle of the gun. This is a fairly easy setup to do quickly, but here are a few precautions when making your rigs. Put your pivot points in your boxes where you want the model to rotate. Do not scale your boxes, unless you scale them at the sub-object level.

For a much more complex version of this rig take a look at Tank_Complete.max. This rig uses dummy objects to modify the damage states. If you have already decided that you really want to modify the model as it takes damage, now if the time to add these to your rig.

Skin that Character Unit

Make sure your biped is in figure mode. Select the model that you want to skin to the rig and add a skin modifier. Click the Add button and add the following biped bones: Bip Pelvis, Bip Thigh, Bip Calf, Bip L Foot, Bip R Thigh, Bip R Calf, Bip R Foot, Bip Spine, Bipd Spine1, Bip L Clavicle, Bip L UpperArm, Bip L Forearm, Bip L Hand, Bip Neck, Bip Head, Bip R Clavicle, Bip R UpperArm, Bip R Forearm, and Bip R Hand.


Select Bones

```
MD
  BIP
    BIP Pelvis
      BIP L Thigh
        BIP L Calf
          BIP L Foot
      BIP R Thigh
        BIP R Calf
          BIP R Foot
      BIP Spine
        BIP Spine1
          BIP L Clavicle
            BIP L UpperArm
              BIP L Forearm
                BIP L Hand
          BIP Neck
            BIP Head
          BIP R Clavicle
            BIP R UpperArm
              BIP R Forearm
                BIP R Hand
  EFFECTS
  LeftFoot
  RightFoot
```

Sort: [✓ Alphabetical] By Type By Color By Size

List Types: ☑ Geometry ☑ Shapes ☑ Lights ☑ Cameras ☑ Helpers ☑ Space Warps ☑ Groups/Assemblies ☑ XRefs ☑ Bone Objects

Selection Sets: Biped

☑ Display Subtree ☐ Case Sensitive ☐ Select Subtree ☐ Select Dependents

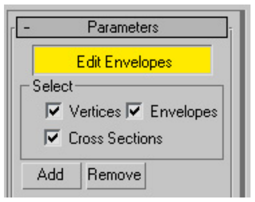
Click on the edit envelopes button and modify your envelopes so that at a glance it looks like the bone you want controlling your vert geometry is overlapping that geometry. Click on your Biped and exit figure mode at this point. You can now watch biped animation, and see how it updates your skinned model. This is another good reason for using our Biped setup, because you get a set of animations that allow you to see what needs to be tweaked on your skin modifier to make your model move properly in all the right areas.

Look for stray verts that are doing bad things, espcially around the shoulders. Don't spend too long tweaking your skin data looking for perfection. It can take a very long time on a simple rig to get correct movement of all your verts on a human. Remember that these guys are fairly small on the screen when playing Civilization 4 and it's more fun playing the game with your new units that worrying about that one part of the back that looks a little weird when the arm is fully extended.

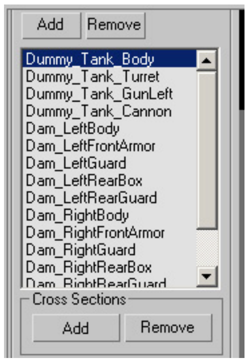
After you are satisfied with the results, you are ready to move on to animating your character.

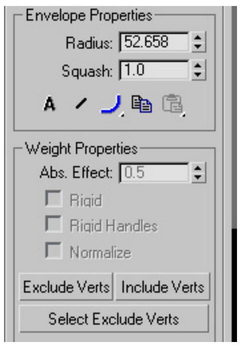
Skin that amazing Mechanical Units

You will begin in the same way as skinning character units. Select your model, add the skin modifier, and from the skin tool bar add the bones you want to use to animate you model. Only select the bones that will be responsible for the animations of this object, and not the bones that you've made to creat damage states with. This will save you time, and you can always add bones after your done the big picture rigging.

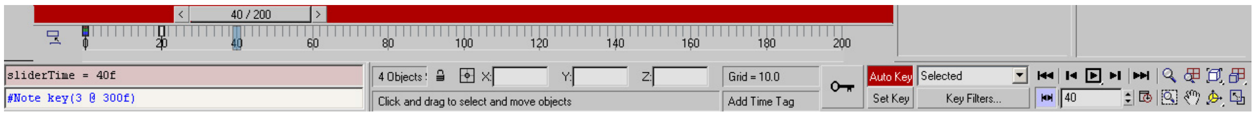


The big difference between skinning character units and mechanical units is that you usually don't have a bunch of bendy parts, and this usually makes skinning mechanical units much quicker. Click the Select Vertices check box under the Skin modifier Parameter menu.





You will then select a bone, a series of verts that you want to follow that bone exclusively, and then type in an Abs. Effect of 1.0 in the Weight properties box. After you have linked all the verts of your model to the appropriate bones, exit the skin modifier mode. Select the bones that you've made, make sure you animation bar is on frame zero and right click on the scrub bar to create a set of position, rotation, and scale keys on frame 0.



```
sliderTime = 40f
#Note key(3 @ 300f)
```

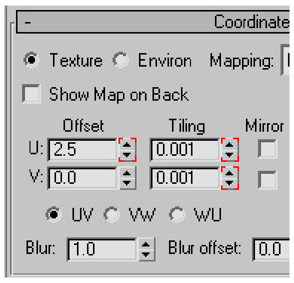
Click and drag to select and move objects

Select these key frames, and move them to frame 20. On frame 40, Slect the Auto Key button on the bottom of the interface and begin animating these bones. Make a quick set of motions over the next 60 frames that should represent the total range of movement needed for your mechanical unit. If your unit has a turret, make sure that the turret can rotate. If you plan of blowing the turret off of the tank, make sure that works as well. This is a set of animations that will be used to test your effectiveness in skinning your mechanical unit, as well as a way to make sure these movements still work after you add your damage bones.

The best way to begin explaining how to rig the stuff for damage state changes to the model is to open up the Tank_Complete.max file at take a look. There are a set of bones attached to the Damage_BodyGeometry Dummy which manipulate the main chasis of the tank. There are a set of bones attached to the Damage_Turret Dummy which manipulates the turret geometry. Finally there is the Damage_TankTexture dummy, which all geometry in the scene is attached to, and that handles the animation of the damaged state texture map, and the appearance of damage smoke.

I used the same timing to animate all the damage states for the mechanical units. From frame 900 to 950 is Damage State 1 and the model should not be damaged. From frame 1000 to 1080 is Damage State 2 and the model should be damaged looking. From frame 1100 to 1180 is Damage State 3 and the model should be very damaged looking. From frame 1200 to 1280 is Damage State 4 will the model should look no different from damage state 3. Damage state 4 incorperates an object or two where smoke is attached, and will appear through an animated visibility track.

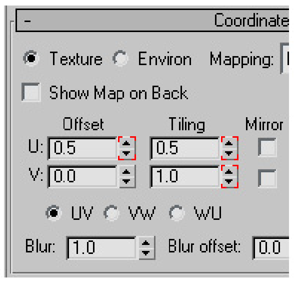
You now want to add the damage state bones to the list of skin bones in the units modifier. Now go to frame 950 and select all the animating damage bones, rightclick on the timing slider and add keyframes for position, rotation, and scale. Use a combination of modifying the skin weights, and animated the damage states to set the right animation keys to make your three damage states. I would do this by importing the damaged looking models into the scene, and just focusing on matching the look of the one through bone animation.


Coordinate
[✓ Texture] Environ Mapping:
☐ Show Map on Back

	Offset	Tiling	Mirror
U:	2.5	0.001	☐
V:	0.0	0.001	☐

[✓ UV] VW WU
Blur: 1.0 Blur offset: 0.0

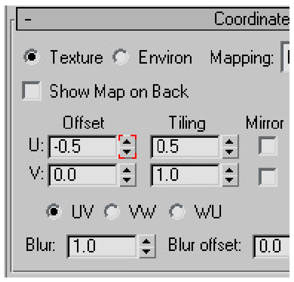
Damage State 1


Coordinate
[✓ Texture] Environ Mapping:
☐ Show Map on Back

	Offset	Tiling	Mirror
U:	0.5	0.5	☐
V:	0.0	1.0	☐

[✓ UV] VW WU
Blur: 1.0 Blur offset: 0.0

Damage State 2


Coordinate
[✓ Texture] Environ Mapping:
☐ Show Map on Back

	Offset	Tiling	Mirror
U:	-0.5	0.5	☐
V:	0.0	1.0	☐

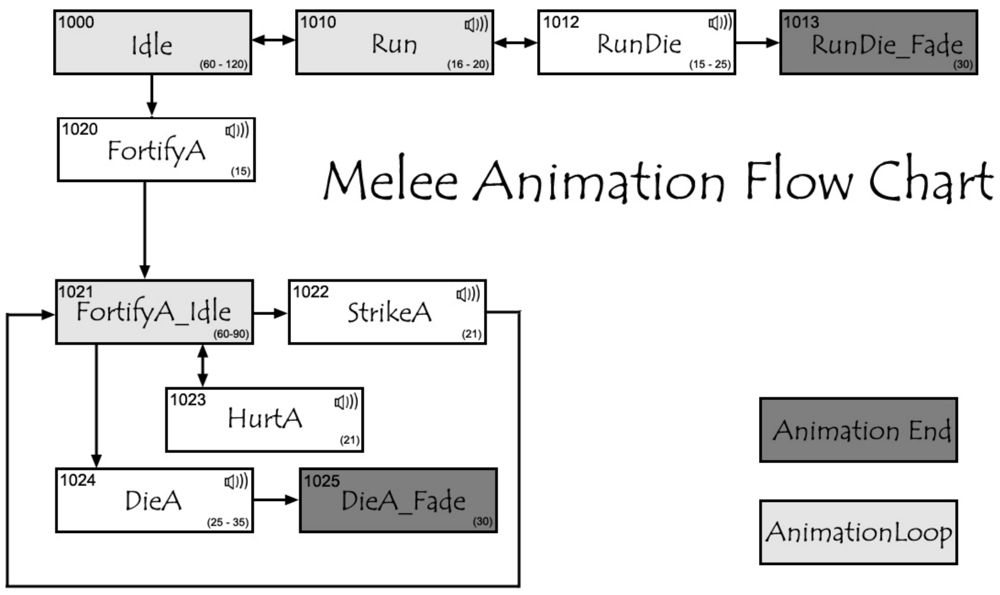
[✓ UV] VW WU
Blur: 1.0 Blur offset: 0.0

Damage State 3 & 4

You will also want to animate the damage state for the texture, and will will do that by modifying the U, V coordinates for the damaged texture in the decal map slot on your material. The damage state 1 numbers are so that no damage will appear by zooming in on the teture to an area where the alpha value is 0.0. The other two values for damage state 2, 3, and 4 come from a 256 x 128 texture value being placed neatly over a 256 x 256 base texture.

Animate your character Unit

Time to make your guy move. The first thing we'll look at is a minimum animation flowchart for a combat unit.



The number in the upper left is the event code for the animation. The number in the lower right is the allowed frame time for your animation. If that number is a frame range, then it's animators choice, if that number is a single number, then make your animation for that sequence that many frames.

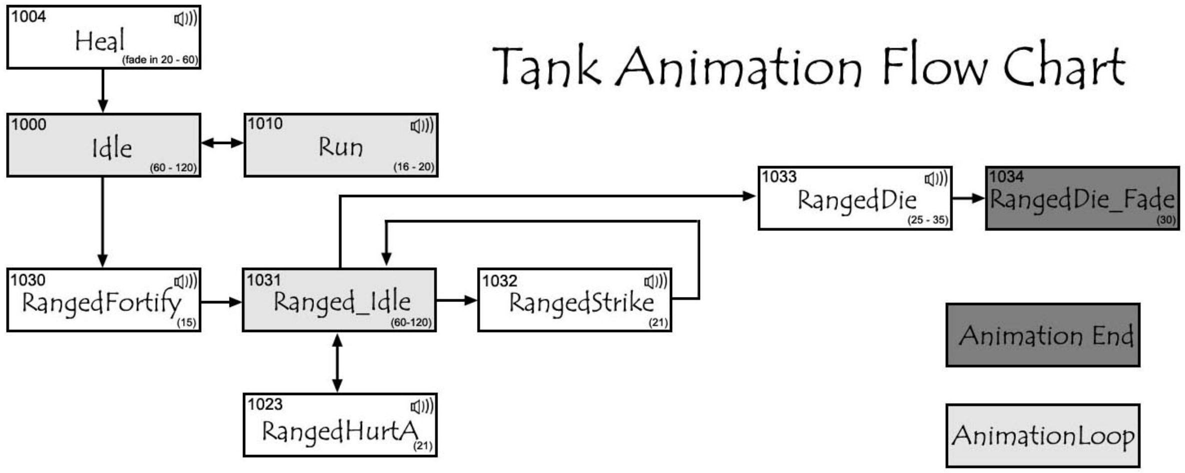
Begin with the Idle pose and animate a simple breathing loop. From there animate a fortify animation where the guy goes into a fortify loop. Animate a quick Strike that returns to the Fortify_Idle. Next, animate a quick Hurt animation. Take the beginning of your Hurt animation, and turn a copy of that into your Die animation. The timing of your strike, hurt, and die animations should have the actual hit occuring at the 11th frame of your animation. Die fade is simply the resting position of the guy fading out. Move on to the run animation now. Finally, do the RunDie animation. This animation is not nearly as important as the rest, because it shows up so rarely in the game.

If you want to do a more animation heavy set, or are doing a settler or worker make sure to check out MotionFlow_Melee.jpg, MotionFlow_Ranged.jpg, MotionFlow_Settler.jpg, and MotionFlow_GreatPeople.jpg pages. Each is sized for printing, and I found it good motivation to print one out while beginning to animate a character so I could cross things off as I progressed.

After your animation set is complete movepat yourself on the back, think of all the CivAddicts that you will be making happy with your hard work, and move on to the next section.

Animate your Mechanical Unit

Time to make your guy move. The first thing we'll look at is a minimum animation flowchart for a combat unit.



Take a look at Animating your Character Unit Section for a description of the above chart as well as advice for beginning you animation. Since mechanical units are so effects heavy I will explain the process of getting effects in the game in the next working section. Our effects creator, Michael Bazzell did an excellent job creating effects for our game, so why not start by using the best available. If you are interested in learning how to make effects of your own, perhaps we can get him to give the Mod community a little love later on this year.

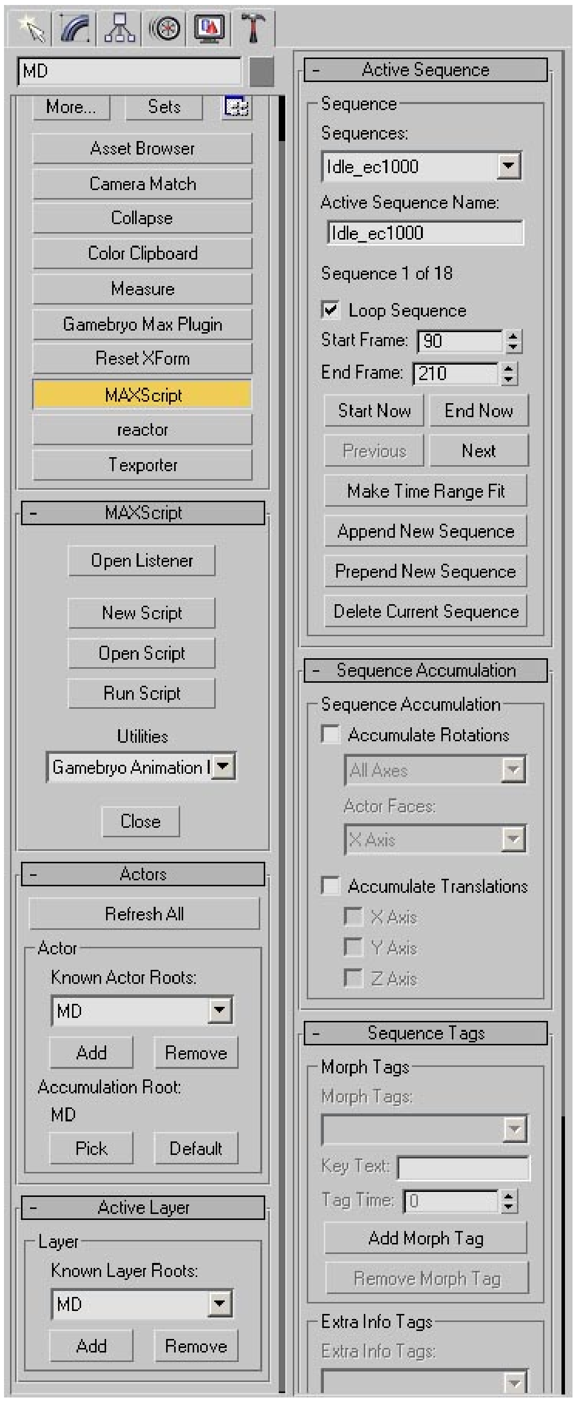
Setting-Up Everything else so your new unit will work in the Game

If you've looked at any of the Complete unit files you will notice there are a bunch of Dummy objects in the scene that we haven't covered at all. These are all in the units for very good reasons, and in this section each of these "helper" dummies will be explained so you can take full advantage of them.

MD

This Helper dummy was initially named Master Dummy but was shortened to MD so that our export file names would be smaler. This is the most important helper dummy in your scene. The top level of your rig, the BIP in your character unit file for example, and all unlinked geometry should be children of this dummy.

By beginning either your character unit or mechanical unit with the base rig we've provied you will save a considerable amount of setup time. The MD has the tags that generate the different event codes that are crucial for getting units animated in Civilization 4. To modify the animation ranges to match your work, select the MD, and click on the utilities Tab on the Command Panel. Select the MAXScript button. A small MAXScript menu will appear under the buttons, and you should select GameBryoANimation Manager from the drop down utilities box.



Object: MD
Utilities: Gamebryo Animation I
Known Actor Roots: MD
Accumulation Root: MD
Known Layer Roots: MD
Sequences: Idle_ec1000
Active Sequence Name: Idle_ec1000
Sequence 1 of 18
[✓] Loop Sequence
Start Frame: 90
End Frame: 210
[ ] Accumulate Rotations
All Axes
Actor Faces: X Axis
[ ] Accumulate Translations
[ ] X Axis
[ ] Y Axis
[ ] Z Axis
Morph Tags:
Tag Time: 0

Here is the control panel where you will set up your animations for use in Civilization 4. The Actor Root is your MD object. In any file you are working on your should only have 1 actor root, any more than that will only confuse the situation.

The active Layer let's you know which level of event code sequences you are tagging. Since the MD is an actor root it is the top layer of your event codes. The only time that we use other layers is to set damage states in mechanical units. You tag the different animation sets by selecting other Layer Roots from this area.

The active sequence is the only area left that we care about. This is where you input the information for each sequence that you have animated. This is a nice interface for adding note tracks to your MD and damage state helpers.

If you have followed my recommendations up to this point you will have a full set of sequences that probably to not line up with your new animations. Delete all the sequences until you have one left, then type in the start frame, then the end frame for your first sequence. Name your sequence whatever your animation is followed with a _ec and the event code number that your animation is. For example, if you have a run sequence from frame 70 to frame 88, you type into the active sequence input are "Run_ec1010". Since the run sequence should loop, you will also take this time to select the loop animation check box.

SOUND

This is the dummy object that stores the note tracks that let the game know what sound to play and when.

Each note track is formatted with the following naming convention: "SOUND:AS3D_UN_LEATHER_CREAK_SHORT_LOW" These correspond to entries in Sid Meiers Civilization 4\Assets\XML\Audio\ Audio3DScripts.xml. This helper's parent needs to be the MD, and under the User Defined properties on this helper should be "NiOptimizeKeep" and "ExtraNoteTrack = 1" to ensure the export of the tags placed on the Effects dummy notetrack.

We would provide the sound engineers animations with frame numbers attached for them to make sounds. These sounds would be added to the game, and to the .xml files. They would then give us a copy of there excel document saved out as a .csv file. Under the extra tag info we could import this .csv file and the Sound Dummy would have all of its sound information automatically generated. We have provided a sample .csv file for you to look at. However you decide to add sound to your unit, the Sound dummy is the way that the game knows what sound to play and when.

EFFECTS

This is the Helper Objects where you define Effects that will be added to your unit by the game. Due to the complexity of things like explosions, and the ability to re-use things like footprints and smoke the effect are not included in the animation of the units. The way that these are added is by a note track tag on the EFFECTS dummy. This helper's parent needs to be the MD, and under the User Defined properties on this helper should be "NiOptimizeKeep" and "ExtraNoteTrack = 1" to ensure the export of the tags placed on the Effects dummy notetrack. All effects are single NIF files with one, non-looping, animation in it. The effects added this way are animated and disapear after playing once.

The process to add an effect begins with creating a dummy object and setting its position and rotation. This specifies the position and direction the effect will be spawned from. The name of the dummy object should begin with Effect_ and then have a name that anyone opening the file should understand. For example if the effect is a black powder blast coming from a rifle it should be name Effect_BlackPowderBlast. Right click on the dummy object that you have now named, select properties from the object, and in the User Defined properties tab type in "NiOptimizeKeep". This ensures that as the export script optimizes your animations down to only what's needed this dummy object is preserved. Also under the User Defined properties add "ExtraNoteTrack = 1".

It is good to merge the Effect you want to use in the scene, and align to the dummy object for creating the effect in game. This probably won't be possible because if you are using our effects you don't have access to the .max file. You'll probably be best off doing a best guess for the alignment, and after testing it in the game, making any changes necessary.

Go to the note track of the EFFECTS dummy and add a notetrack at the time you want the desired effect to play in your animation. The note track should have text formatted as follows: EFFECT: NameOfYourDummy:NameOfTheEffectScript. For added clarity, let's assume the name of your dummy object is the one we name above, and the script name that loads the effect you want is called EFFECT_BLACKPOWDER the note track would read: EFFECT:Effect_BlackPowderBlast:EFFECT_BLACKPOWDER. All the effect links are in Sid Meiers Civilization 4\Assets\Xml\Misc\CIV4EffectInfos.xml. They are specified by an effect name ("Type") and the path to the art file.

```
-<EffectInfo>
     <Type>EFFECT_MININGSPARK</Type>
     <Description>Hot Sparks</Description>
     <fScale>1.0</fScale>
     <fUpdateRate>1.0</fUpdateRate>
     <Path>Art/Effects/work_mine/MiningSpark.nif</Path>
     <bIsProjectile>0</bIsProjectile>
  </EffectInfo>
```

Now it’s time to export your animation and make sure the effect that you’ve taken great care to add actually works. You want to add your newly exported animations to the appropriate folder of a running version of Civ4. You also want to check the CIV4EffectInfos.xml under the \XML\Misc directory and make sure the effect script you are calling exists. If everything is 100% you should now be able to see it in the game, and you rock.

ATTACHABLES

This is the dummy object that stores the data in its User Defined Properties of that lets the game know what effects to attach that can be played as a loop. Common uses for looped effects are for damage smoke, or engine smoke that appears during movement. This helper’s parent needs to be the MD, and under the User Defined properties on this helper should be "NiOptimizeKeep".

Creating Attachables are very similar to adding effects except that attachables will be attached as part of the model when loaded in Civilization 4. Attachables can be added to Sid Meiers Civilization 4Assets\Xml\Misc\CIV4AttachableInfos.xml by specifying an attachable name and the path to the art file. All attachables need to be single .NIF file with either a looped animation or no animation present.

To add an Attachable create a box object with the correct position and orientation where you want the attachable attached as a child (the reason to make the object geometry is because you can animate its visibility track-which you can't animate if the object is a dummy). Repeat this step for as many attachables as are in the scene.

In the user defined section of the ATTACHABLES dummy add a line following this format: “ATTACH:NameOfYourDummy:NameOfTheAttachable”, for example: “ATTACH:SmokeNode:ATTACHABLE_SMOKE”. Repeat this step each of the attachables that you want on the unit.

Make sure that your attachable geometries also have NiOptimizeKeep in their user defined properies. Next step is to animate the visibility tracks of the attachable geometries. Make sure that they are visible in the first frame of the animation (good practice for all objects with visibility animations). Check that the object has a visibility value of 1.0 on the frames that you want the attachable to be seen, and that the visibility value is 0.0 on the tracks where you want the attachable to be hidden. Export and test your work so far. You should see your Attachable Geometry animating with its visibility track when you want it to appear and dissapear.

Now it's time for the last optimization step, and before you start make sure all your objects, especially the attachable geometry placers are named clearly. Add an edit mesh or edit poly to your attachable geometry. Select and delete all geometry within this object. This gives you a node with animation tracks, without the overhead of storing extra geometry. It does make the file slightly harder to understand at first glance, but should be worth it to rid ourselves of all that extra data floating around.

Test once again in the game, and make sure the effect is working. Give yourself a pat on the back knowing you how to get the job done, or as Elvis would say, for Taking Care of Business⚡!

That's it for the Instructions. What follows is an internal document we wrote up for optimizing our finished work so we wouldn't clog the heart of the game engine with cholesterol filled units. Have fun Modding, and thanks for loving our game enough to add to it.

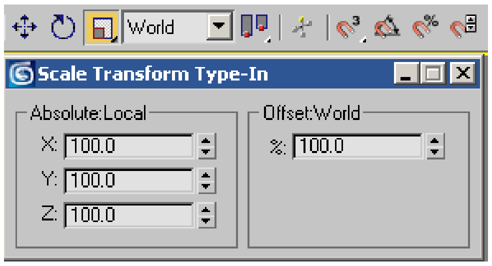
Dorian Newcomb March 2, 2006

Unit Optimizations

Object Check

Make sure there are not scale transforms on any of you models in the scene. We all know non-Uniform scaling is terrible, but units that have a uniform scale applied to them at frame 0 are just as jacked.

Easiest way to check is to right click on the scale button for each object in the scene. Make sure your reference coordinate system is set to world. If you see anything other than this…



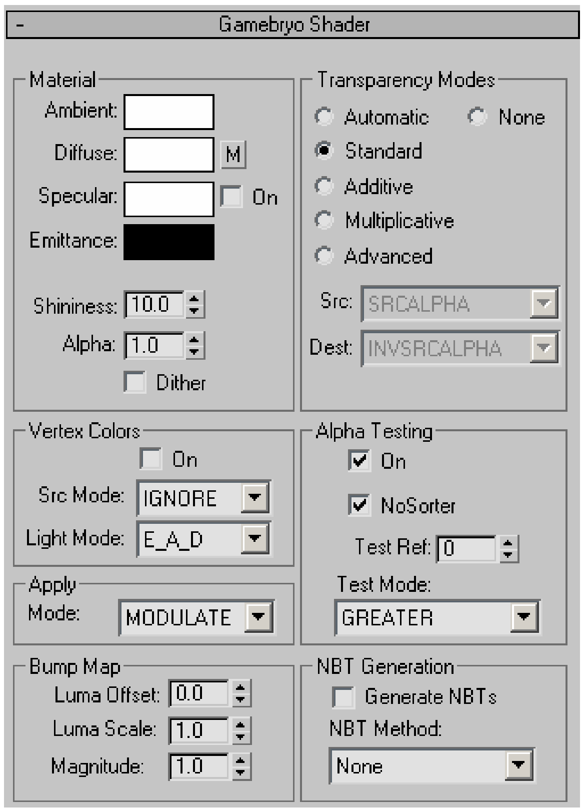
X: 100.0
Y: 100.0
Z: 100.0
%: 100.0

You need to fix it. If the object is a non skinned animated object there is an easy fix that will save time. Make sure the non 100.0 scale numbers are uniform (this will not work if your numbers are X [56.33], Y [75.00], Z [75.00] because this is non-uniform scale which sucks!). Animate the first frame frame so the scale is 100 across the boards and you are done. If this is a skinned object you will need to reset your x-form, or do what I like to do, which is to create a box, set to [0, 0, 0], make it an editable mesh, and attach the old "jacked" geometry to your new pure box. Either way you go, you will have to reskin, which can be a time sink, but it is important for performance.

Make sure to check for each object.

Material Check

Here's how to set up your material so that it rocks.



Gamebryo Shader

Material
Ambient: □
Diffuse: □ M
Specular: □ ☐ On
Emittance: ■
Shininess: 10.0
Alpha: 1.0
☐ Dither

Transparency Modes
Automatic [✓ Standard] Additive Multiplicative Advanced None
Src: SRCALPHA
Dest: INVSRCALPHA

Vertex Colors
☐ On
Src Mode: IGNORE
Light Mode: E_A_D

Alpha Testing
☑ On
☑ NoSorter
Test Ref: 0
Test Mode: GREATER

Apply
Mode: MODULATE

Bump Map
Luma Offset: 0.0
Luma Scale: 1.0
Magnitude: 1.0

NBT Generation
☐ Generate NBTs
NBT Method: None

Things of note: Ambient, diffuse, and specular are all [255, 255, 255] and emittance is [0, 0, 0]. This is what you want almost all the time, certain excetptions occure on objects that are billboarded (why light a billboard object, so usually the emittance will be set to [255, 255, 255] on those). Make sure vertex color check box is unchecked with Source Mode set to Ignore and Light Mode set to E_A_D. Alpha Testing should be on, with NoSorter checked as well. If you find these setting are screwing stuff, find out which settings work, and get approval for the changes from either Tom or Bart.

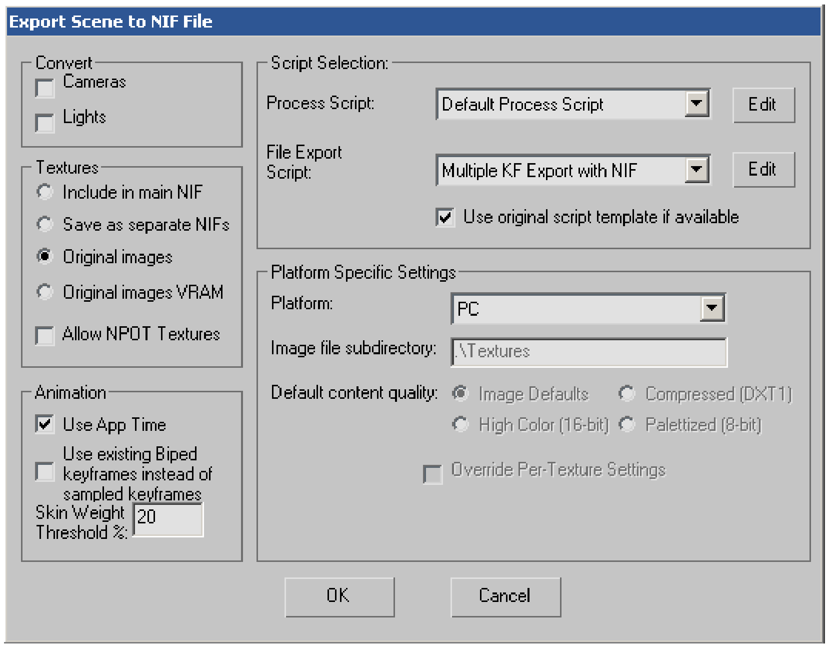
Make sure you spend the time to name your textures a meaningful name. Naming conventions whould be as follows (these names will help when it comes time to make the FX version of the file).

UnitName_Skinned_TeamColor
UnitName_Skinned_NonTeamColor
UnitName_Case1_NotSkinned
UnitName_Case2_NotSkinned

This should handle most units, and if you find yourself unsure, go with your instincts. For the most part, these names will make you final steps of exporting easier, and will make it so the expansion team does not whisper curses under their breath about any of us.

Set up your export stuff

These are the options I use when exporting character and mechanical units.


Export Scene to NIF File

Convert
- [ ] Cameras
- [ ] Lights

Textures
- ○ Include in main NIF
- ○ Save as separate NIFs
- ● Original images
- ○ Original images VRAM
- [ ] Allow NPOT Textures

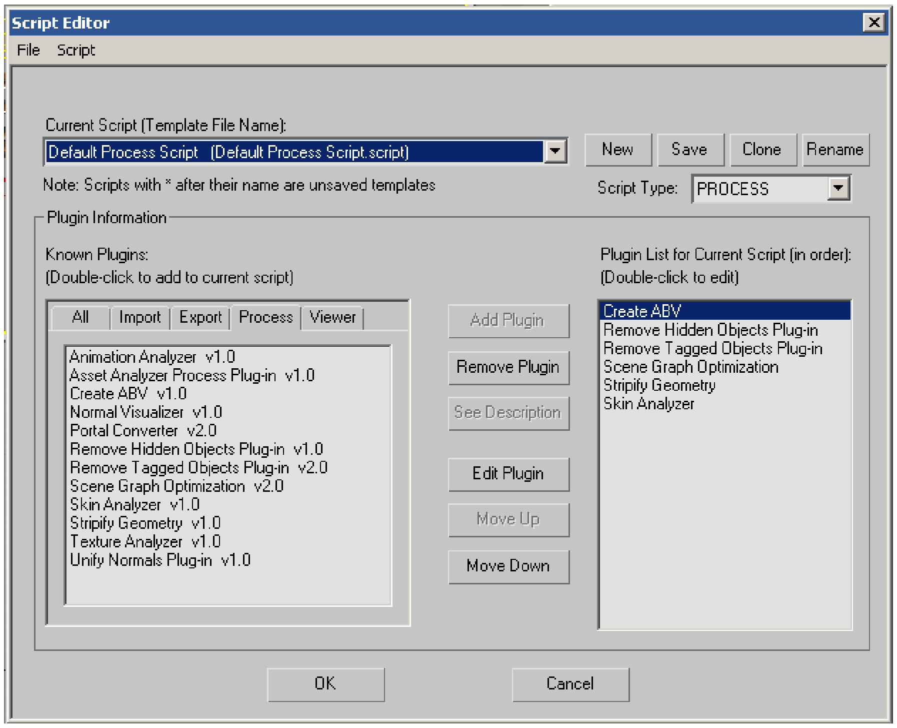
Animation
- [✓] Use App Time
- [ ] Use existing Biped keyframes instead of sampled keyframes

Skin Weight Threshold %: 20

Script Selection:
Process Script: Default Process Script
File Export Script: Multiple KF Export with NIF
- [✓] Use original script template if available

Platform Specific Settings
Platform: PC
Image file subdirectory: .\Textures
Default content quality: ● Image Defaults ○ Compressed (DXT1) ○ High Color (16-bit) ○ Palettized (8-bit)
- [ ] Override Per-Texture Settings

The 3 exceptions to this setup are as follows
If you have an environment light, you obvious want to convert your lights.
The damage state for certain tanks use a skin weight threshold below 20% so I can get things distored on the entire unit and still maintain the animation I've already done.
Finally, there are many times when you will want to not stripify, so the default process script may be the default without stripification. Unforturnately to see if your model is improved with or without stripification, you need to export both ways and comare the two files. This leads us to the default process script which we will also tweak for performance.


Script Editor

Current Script (Template File Name): Default Process Script (Default Process Script.script)
Note: Scripts with * after their name are unsaved templates
Script Type: PROCESS

Plugin Information
Known Plugins:
(Double-click to add to current script)

Animation Analyzer v1.0
Asset Analyzer Process Plug-in v1.0
Create ABV v1.0
Normal Visualizer v1.0
Portal Converter v2.0
Remove Hidden Objects Plug-in v1.0
Remove Tagged Objects Plug-in v2.0
Scene Graph Optimization v2.0
Skin Analyzer v1.0
Stripify Geometry v1.0
Texture Analyzer v1.0
Unify Normals Plug-in v1.0

Plugin List for Current Script (in order):
(Double-click to edit)

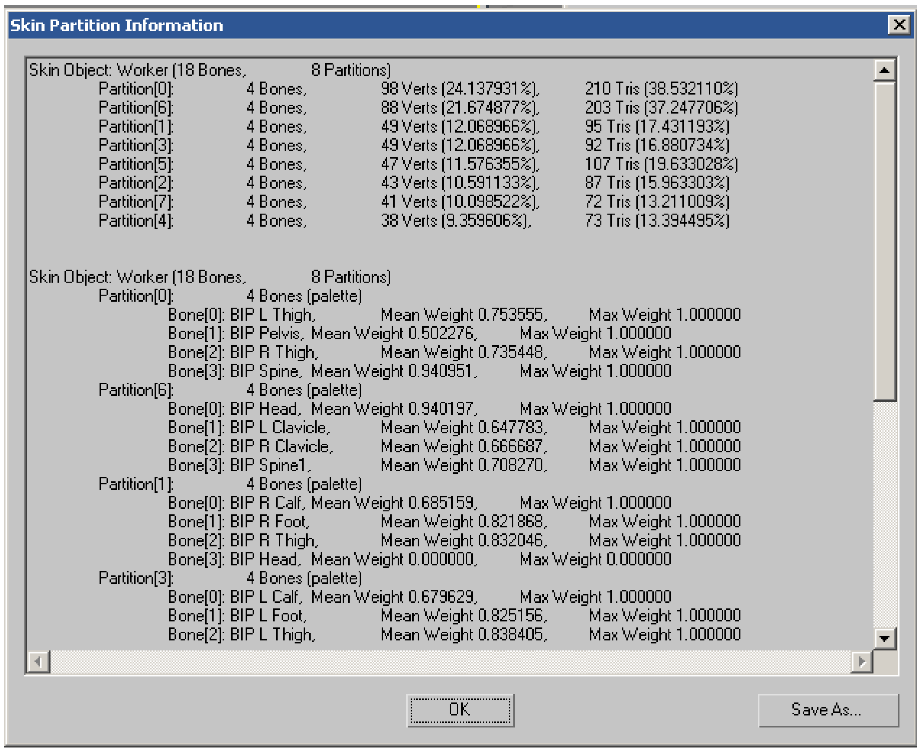
Create ABV
Remove Hidden Objects Plug-in
Remove Tagged Objects Plug-in
Scene Graph Optimization
Stripify Geometry
Skin Analyzer

There are 3 additional plug-in we need to make sure we are using. The order they are in is important, so make sure your default plug-in list is the same.

Remove Hidden Objects Plug-in – Any object that is hidden upon export is deleted. If the object is important to animated objects parented to the object, the object is kept. I refer to this kind of removal as a “soft” object removal.

Remove Tagged Objects Plug-in - Any object that has the tag “NiRemoveObject” will be deleted. If the object is important to animated objects parented to the object, the object is removed as well as all its children.

Skin Analyzer – Upon export a screen pops up to let you know how efficient your skinned objects are attached to you animations. The pop-up looks like this…

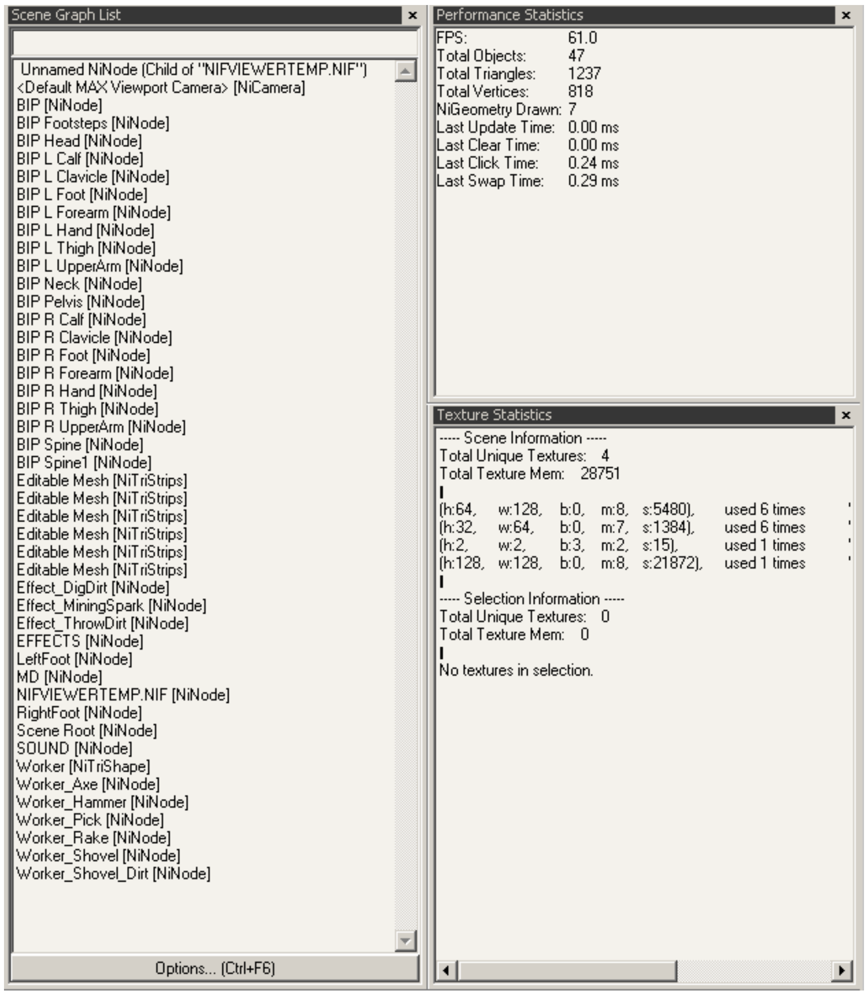


```
Skin Object: Worker (18 Bones,          8 Partitions)
          Partition[0]:       4 Bones,        98 Verts (24.137931%),     210 Tris (38.532110%)
          Partition[6]:       4 Bones,        88 Verts (21.674877%),     203 Tris (37.247706%)
          Partition[1]:       4 Bones,        49 Verts (12.068966%),     95 Tris (17.431193%)
          Partition[3]:       4 Bones,        49 Verts (12.068966%),     92 Tris (16.880734%)
          Partition[5]:       4 Bones,        47 Verts (11.576355%),     107 Tris (19.633028%)
          Partition[2]:       4 Bones,        43 Verts (10.591133%),     87 Tris (15.963303%)
          Partition[7]:       4 Bones,        41 Verts (10.098522%),     72 Tris (13.211009%)
          Partition[4]:       4 Bones,        38 Verts (9.359606%),      73 Tris (13.394495%)


Skin Object: Worker (18 Bones,          8 Partitions)
          Partition[0]:       4 Bones (palette)
                    Bone[0]: BIP L Thigh,        Mean Weight 0.753555,     Max Weight 1.000000
                    Bone[1]: BIP Pelvis,  Mean Weight 0.502276,     Max Weight 1.000000
                    Bone[2]: BIP R Thigh,        Mean Weight 0.735448,     Max Weight 1.000000
                    Bone[3]: BIP Spine,  Mean Weight 0.940951,     Max Weight 1.000000
          Partition[6]:       4 Bones (palette)
                    Bone[0]: BIP Head,  Mean Weight 0.940197,     Max Weight 1.000000
                    Bone[1]: BIP L Clavicle,     Mean Weight 0.647783,     Max Weight 1.000000
                    Bone[2]: BIP R Clavicle,     Mean Weight 0.666687,     Max Weight 1.000000
                    Bone[3]: BIP Spine1,         Mean Weight 0.708270,     Max Weight 1.000000
          Partition[1]:       4 Bones (palette)
                    Bone[0]: BIP R Calf, Mean Weight 0.685159,     Max Weight 1.000000
                    Bone[1]: BIP R Foot,         Mean Weight 0.821868,     Max Weight 1.000000
                    Bone[2]: BIP R Thigh,        Mean Weight 0.832046,     Max Weight 1.000000
                    Bone[3]: BIP Head,  Mean Weight 0.000000,     Max Weight 0.000000
          Partition[3]:       4 Bones (palette)
                    Bone[0]: BIP L Calf,  Mean Weight 0.679629,     Max Weight 1.000000
                    Bone[1]: BIP L Foot,         Mean Weight 0.825156,     Max Weight 1.000000
                    Bone[2]: BIP L Thigh,        Mean Weight 0.838405,     Max Weight 1.000000
```

Remove Those Bones

Now it’s time to remove totally unessecary bones, and maybe even bones you think may be important, but can get rid of. As part of this phase, I’ll include how to interpret what our .nif file can tell us.



```
Unnamed NiNode (Child of "NIFVIEWERTEMP.NIF")
<Default MAX Viewport Camera> [NiCamera]
BIP [NiNode]
BIP Footsteps [NiNode]
BIP Head [NiNode]
BIP L Calf [NiNode]
BIP L Clavicle [NiNode]
BIP L Foot [NiNode]
BIP L Forearm [NiNode]
BIP L Hand [NiNode]
BIP L Thigh [NiNode]
BIP L UpperArm [NiNode]
BIP Neck [NiNode]
BIP Pelvis [NiNode]
BIP R Calf [NiNode]
BIP R Clavicle [NiNode]
BIP R Foot [NiNode]
BIP R Forearm [NiNode]
BIP R Hand [NiNode]
BIP R Thigh [NiNode]
BIP R UpperArm [NiNode]
BIP Spine [NiNode]
BIP Spine1 [NiNode]
Editable Mesh [NiTriStrips]
Editable Mesh [NiTriStrips]
Editable Mesh [NiTriStrips]
Editable Mesh [NiTriStrips]
Editable Mesh [NiTriStrips]
Editable Mesh [NiTriStrips]
Effect_DigDirt [NiNode]
Effect_MiningSpark [NiNode]
Effect_ThrowDirt [NiNode]
EFFECTS [NiNode]
LeftFoot [NiNode]
MD [NiNode]
NIFVIEWERTEMP.NIF [NiNode]
RightFoot [NiNode]
Scene Root [NiNode]
SOUND [NiNode]
Worker [NiTriShape]
Worker_Axe [NiNode]
Worker_Hammer [NiNode]
Worker_Pick [NiNode]
Worker_Rake [NiNode]
Worker_Shovel [NiNode]
Worker_Shovel_Dirt [NiNode]
```

```
FPS:                61.0
Total Objects:      47
Total Triangles:    1237
Total Vertices:     818
NiGeometry Drawn:   7
Last Update Time:   0.00 ms
Last Clear Time:    0.00 ms
Last Click Time:    0.24 ms
Last Swap Time:     0.29 ms
```

```
----- Scene Information -----
Total Unique Textures:   4
Total Texture Mem:   28751
|
(h:64,    w:128,   b:0,   m:8,   s:5480),     used 6 times
(h:32,    w:64,    b:0,   m:7,   s:1384),     used 6 times
(h:2,     w:2,     b:3,   m:2,   s:15),       used 1 times
(h:128,   w:128,   b:0,   m:8,   s:21872),    used 1 times
|
----- Selection Information -----
Total Unique Textures:   0
Total Texture Mem:   0
|
No textures in selection.
```

Performance Statistics are very useful. The total number of objects includes both NiTriStrips and Nodes. As far as Gamebryo is concerned for each real object, say a box, the NiTriStrips hold the geometry data of an obejct and the NiNode holds the position, rotation, and scale information. Because we hide the Biped Geometry on export, Gamebryo only holds the NiNodes for each bone. Each unique piece of geometry adds 2 objects to this total. Unique geometry is also created whenever an object has more than one material assigned to it, so something may look really clean in 3dMax, but spit out a large amount of objects. Other things that add to the object count are MD, SOUND, EFFECTS, and all the effect nodes we use to generate effects. For most character units it is good to have this number be < 30 (18 bone nodes, 4 for character and weapon, 6 for effects, sound, footsteps, and the MD). This is a worker file I'm using as an exmple and he has 5 extra tools, as well as a few extra effect nodes which raises this number to 47. Obviously mounted units will be about twice the recommended size, still aiming at being < 60.

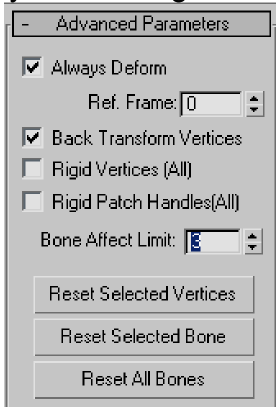
The total number of NiGeometries should be around 2. I've been told as long as there are only two at any given time that's still great. Since this is the case with the worker, even though there are 7 NiGeometry Draws, functionally speaking in the game it is usually 1 (just the worker) or 2 (the worker and his axe). If you have a large number here you have a problem.

Our units should not be using toes or fingers in the skin data. We should also not be using the bip either. That leaves us with 19 bones (7 in the pelvis and legs / 4 in the spine and head/ 8 in the arms). At this point I strongly recommend removing the neck from your bone list. Because the units are so low, there is not enough nuiance of form to get a lot of mile-age out of keeping the neck. I have also found that usually elimiating the Clavicles does not compromise the animation and geometry too badly. There are certainly times to keep these, and if you find yourself spending too much time fixing you skin data after removing these, it may be worthwhile just keeping them. I have also deleted the Spine bone for missionaries because they don't need them. The point is to remove anything you can at this point in your bones because each bone adds so much extra stuff. At the worst each piece of unique geometry that is skinned needs to use < 19 bones. The toughest part of meeting this goal will be the horse units. I will take a look at the knight's horse, and see if I can get the count down this low.

Check the number of Partitions

This is the greatest performance pinch on non shader cards, and is the one thing as animators that we have the most control over. On non shader cards only 4 bones can control a single object. Our objects have 18 bones, so in theory, that could be broken up into 5 objects. In reality though the 18 bones translate into 12 object, and the process of breaking the single object into multiple objects is call partitioning. For each partition, a separate game draw needs to be called, so instead of drawing **6 objects** for 3 warriors (3 warriors / 3 clubs), there are **39 draws** needed (36 for the 3 warriors with 12 partitions each / 3 clubs). This is where Tom Whittaker's FX shader really saves us (1 draw for each unique object with up to 19 bones), but our minimal spec needs us to respect it's limitations. With tthat in mind, I think it is a reasonable goal to have < 9 partitions total for each unit we make, double that number for mounted units. The best way to make sure your partitions are as few as needed follows…

Limit the number of bones you are using (we've gone over that already, but it's important).
Make sure you skin data is clean (don't have sloppy ass envelopes where the head is affecting half your object, but not enough to truly notice. You can also set a bone affect limit in you skin modifyer. It is found under the advanced parameters, and you're probably fine setting this number to 3.


- Advanced Parameters

[✓] Always Deform

Ref. Frame: 0

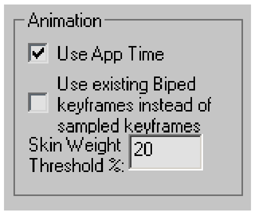
[✓] Back Transform Vertices

[ ] Rigid Vertices (All)

[ ] Rigid Patch Handles(All)

Bone Affect Limit: 3

You can also increase the skin weight threshold % to 25 or even 30%.

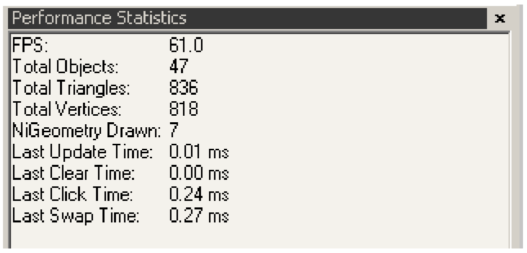


```
Animation
[✓] Use App Time
[ ] Use existing Biped keyframes instead of sampled keyframes
Skin Weight Threshold %: 20
```

Please be careful and check what that is doing to the look of your animation, because is you are not careful, it can truly destroy what was once a beautifully skinned object.

Export and Check

It is also a good time to look at total triangle and total geometry stuff and see if you get a large drop by not stripifying. This unit gets a huge drop by not stripifying, here are the actual numbers.



```
Performance Statistics
FPS:               61.0
Total Objects:     47
Total Triangles:   836
Total Vertices:    818
NiGeometry Drawn:  7
Last Update Time:  0.01 ms
Last Clear Time:   0.00 ms
Last Click Time:   0.24 ms
Last Swap Time:    0.27 ms
```

Stripifying adds about 50% more Triangles, so I would decide not to stripify this file.

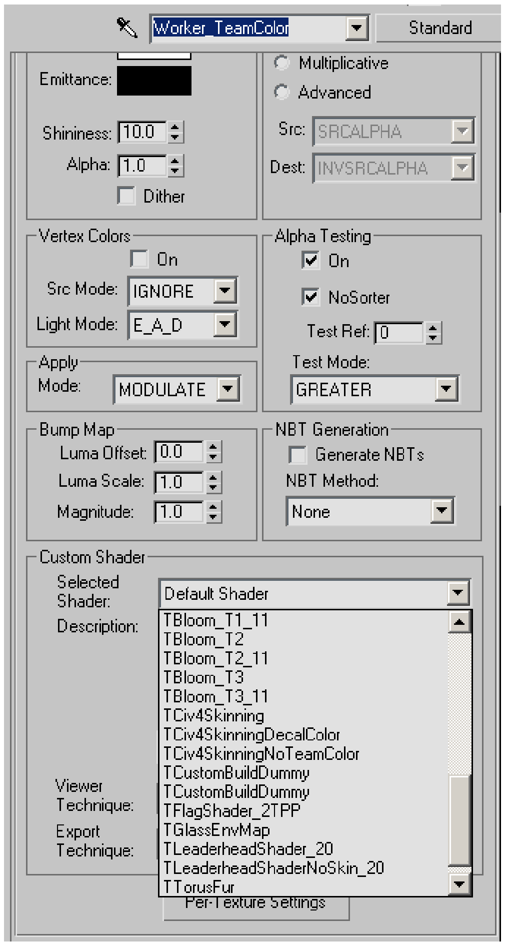
Save and export out your final *Unit.max* and your final *Unit.nif and Unit.kfm*

You are looking to make sure that your .max file is nice and clean and that when other people open it up everything will work. The settings that you choose when exporting your file will be save with your .max file so the final .max file is the file you save after your .nif and .kfm are 100%. Things to look for that are simple errors in your export…

Make sure your die fades and heals have the right Alpha animation.
Check you loops and make sure there is no jumping.

Create your FX shader Version of the File

Remember when I recommended naming your materials like this…
UnitName_Skinned_TeamColor
UnitName_Skinned_NonTeamColor
UnitName_Case1_NotSkinned
UnitName_Case2_NotSkinned
Well now is the time to make the FX version of your export and your .max file. This is basically a texture change with a shader applied to it.


Worker_TeamColor | Standard

Emittance:

Shininess: 10.0

Alpha: 1.0

Dither

Multiplicative

Advanced

Src: SRCALPHA

Dest: INVSRCALPHA

Vertex Colors

On

Src Mode: IGNORE

Light Mode: E_A_D

Apply Mode: MODULATE

Alpha Testing

[✓ On]

[✓ NoSorter]

Test Ref: 0

Test Mode: GREATER

Bump Map

Luma Offset: 0.0

Luma Scale: 1.0

Magnitude: 1.0

NBT Generation

Generate NBTs

NBT Method: None

Custom Shader

Selected Shader: Default Shader

Description:

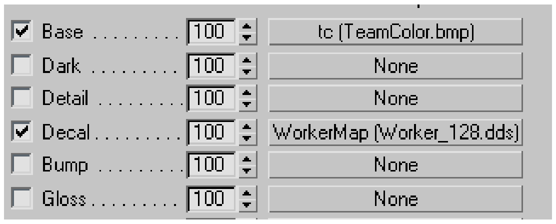
Viewer Technique:

Export Technique:

TBloom_T1_11
TBloom_T2
TBloom_T2_11
TBloom_T3
TBloom_T3_11
TCiv4Skinning
TCiv4SkinningDecalColor
TCiv4SkinningNoTeamColor
TCustomBuildDummy
TCustomBuildDummy
TFlagShader_2TPP
TGlassEnvMap
TLeaderheadShader_20
TLeaderheadShaderNoSkin_20
TTorusFur

Per-Texture Settings

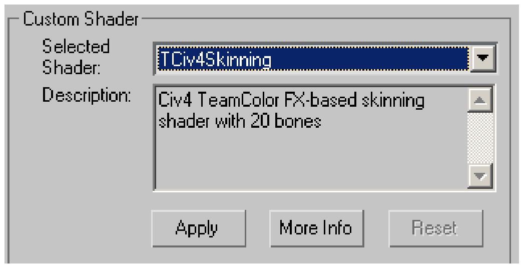
In the material area you can see the pull down custom shader menu. The three that we will be using are the TCIV4Skinning, TCIV4SkinningNoTeamColor, and TCIV4SkinningDecalColor. What follows are the changes you make to each texture, assuming you named your stuff properly.



✓ Base	100	tc (TeamColor.bmp)
Dark	100	None
Detail	100	None
✓ Decal	100	WorkerMap (Worker_128.dds)
Bump	100	None
Gloss	100	None

UnitName_Case1_NotSkinned: A object that is not skinned. Do nothing to this material.

UnitName_Skinned_TeamColor: A skinned object with team color. Drag the decal texture into the base material and drag none into the decal slot. Select TCiv4Skinning from the Selected Shader Drop down Menu. After you do that, make sure you press the apply button at this point.



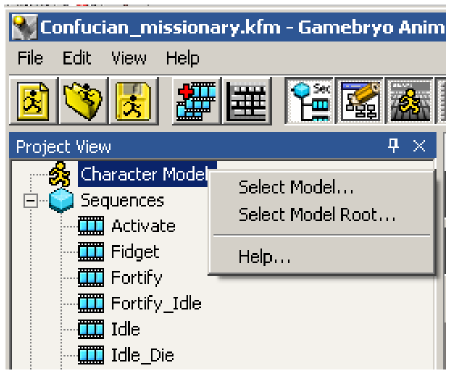
If you do not hit apply it doesn't apply the shader to your material, so nothing will work.

UnitName_Skinned_NonTeamColor: A skinned object without team color. Drag the decal texture into the base material and drag none into the decal slot. Select TCiv4SkinningNoTeamColor from the Selected Shader Drop down Menu. After you do that, make sure you press the apply button at this point. If you do not hit apply it doesn't apply the shader to your material, so nothing will work.

Special Unit with Damage States and an animated Composite Material:
This does not work in the game, so don't apply TCiv4DecalColor to anything. Hopefully I will update this section with good news.

Export out your FX shader version of the File

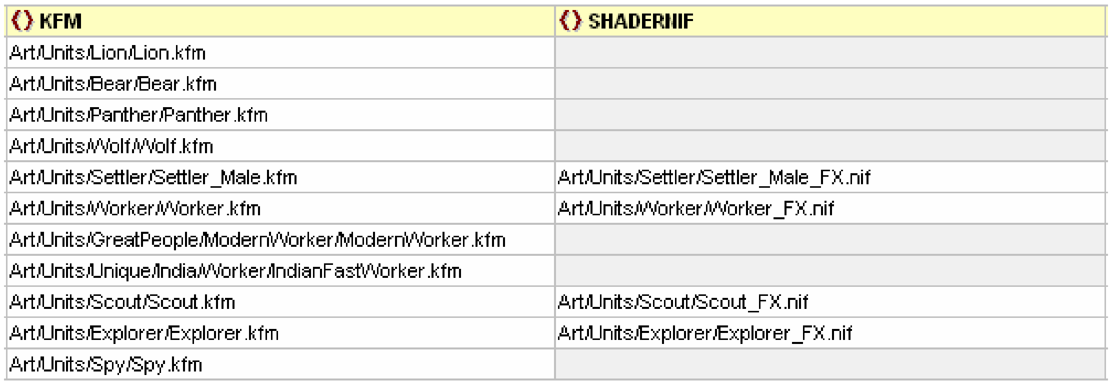
You'll want to export out this new file the same way you exported out the previous file. Now it's time to test your work and make sure this new version works. Open up the .kfm of the original file. Right click on the Character Model and load the _FX.nif version of your file.



All of the geometry that is using the shader will now cease from being lit (it will look black). Even with this limitation you should be able to see if everything is animating correctly. Fix the file if there are any problems, if not you are good to good.

Add to Perforce, Modify the ArtDefines_Unit.xml, Test, and Submit

It's time to submit everything to P4. You will want to add your 2 max files, the master max file and the FX max file. You will also add all of the base export files. The only _FX export you add will be the .nif. Take this time to open up the projects/main/civilization4/assets/XML/Art/ CIV4ArtDefines_Unit.xml file. You need to add an entry in the .xml so that the game will use your _FX version.



KFM	SHADERNIF
Art/Units/Lion/Lion.kfm	
Art/Units/Bear/Bear.kfm	
Art/Units/Panther/Panther.kfm	
Art/Units/Wolf/Wolf.kfm	
Art/Units/Settler/Settler_Male.kfm	Art/Units/Settler/Settler_Male_FX.nif
Art/Units/Worker/Worker.kfm	Art/Units/Worker/Worker_FX.nif
Art/Units/GreatPeople/ModernWorker/ModernWorker.kfm	
Art/Units/Unique/India/Worker/IndianFastWorker.kfm	
Art/Units/Scout/Scout.kfm	Art/Units/Scout/Scout_FX.nif
Art/Units/Explorer/Explorer.kfm	Art/Units/Explorer/Explorer_FX.nif
Art/Units/Spy/Spy.kfm	

Make sure you check in your modified CIV4ArtDefines_Unit.xml file.

Before you submit this batch of new files to Perforce, copy the unit and xml over to your game folder, and test in the game. If the link for the _FX file is bad it will cause the game to have an assert error.

Cheat Sheet

Object Check
No scale on first frame

Material Check
Ambient & Diffuse [255, 255, 255] Emittance [0, 0, 0]
No Vertex Color Alpha testing on with NoSorter on
Naming Convention
UnitName_Skinned_TeamColor
UnitName_Skinned_NonTeamColor
UnitName_Case1_NotSkinned

Set up your export stuff
Default or Default without Strip – Remove Hidden, Removed Tagged, Skin Partition Information

Remove Those Bones
No fingers, toes, bip, or nubs
Consider no Neck and no clavicles
If you can get away with it… drop a spine
Check your geometries, can you get rid of any
< 20 for characters
< 40 total for mounted units with <20 for animal <20 for human

Check the number of Partitions
<9 is the goal

Export and Check
Check the stripification savings. If more that 30% more goemetry is being created, don't strip.

Save and export out your final *Unit.max* and your final *Unit.nif and Unit.kfm*
Check your fades and object visibilities. Check that your note tags are showing up as well.

Create your FX shader Version of the File
Swap over your textures

UnitName_Skinned_TeamColor	>>>	TCiv4Skinning
UnitName_Skinned_NonTeamColor	>>>	TCiv4SkinningNoTeamColor
UnitName_Case1_NotSkinned	>>>	Do nothing

Export out your FX shader version of the File
Test the Unit_FX.nif in the Unit.kfm file

Add to Perforce, Modify the ArtDefines_Unit.xml, Test, and Submit
Add unit_FX.nif to Artdefines_Unit.xml
Test new unit and new .Artdefines_Unit.xml in game
Submit with pride!

\\civ4-build\civ4\AnalyzeLogs

A good area to check to find out how optimized your units are.