Advanced Modding Manual

By Ino-Co Plus LLC (2014)

v.1.0

Table of Contents

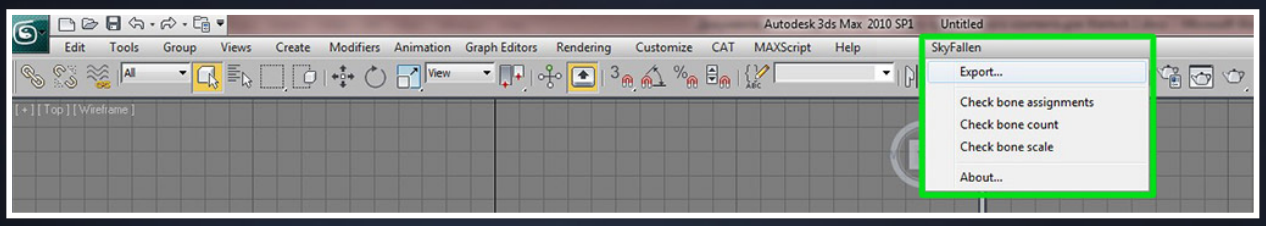
How to prepare and export graphical assets for Warlock 2

3D

To be able to export any 3D models, one has to copy the BMExport2010.dle into the folder 'Plugins' in 3ds Max 2010. The libraries BMCore_Debug.dll, msvcp110d.dll, msvcr110d.dll should be copied to the main folder of 3ds Max 2010.

Note that only the 32-bit version of 3ds Max 2010 is officially supported by the exporter plugin!

The next time 3ds Max 2010 is started, the new export menu for SkyFallen engine models will appear on the main screen:

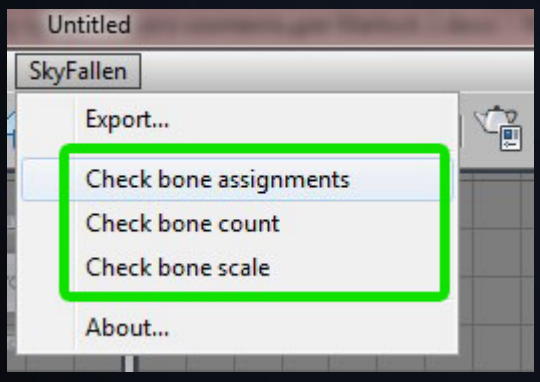


Remember! All the names of every asset in the game must be unique!

Unit models

Technical 3D model requirements:

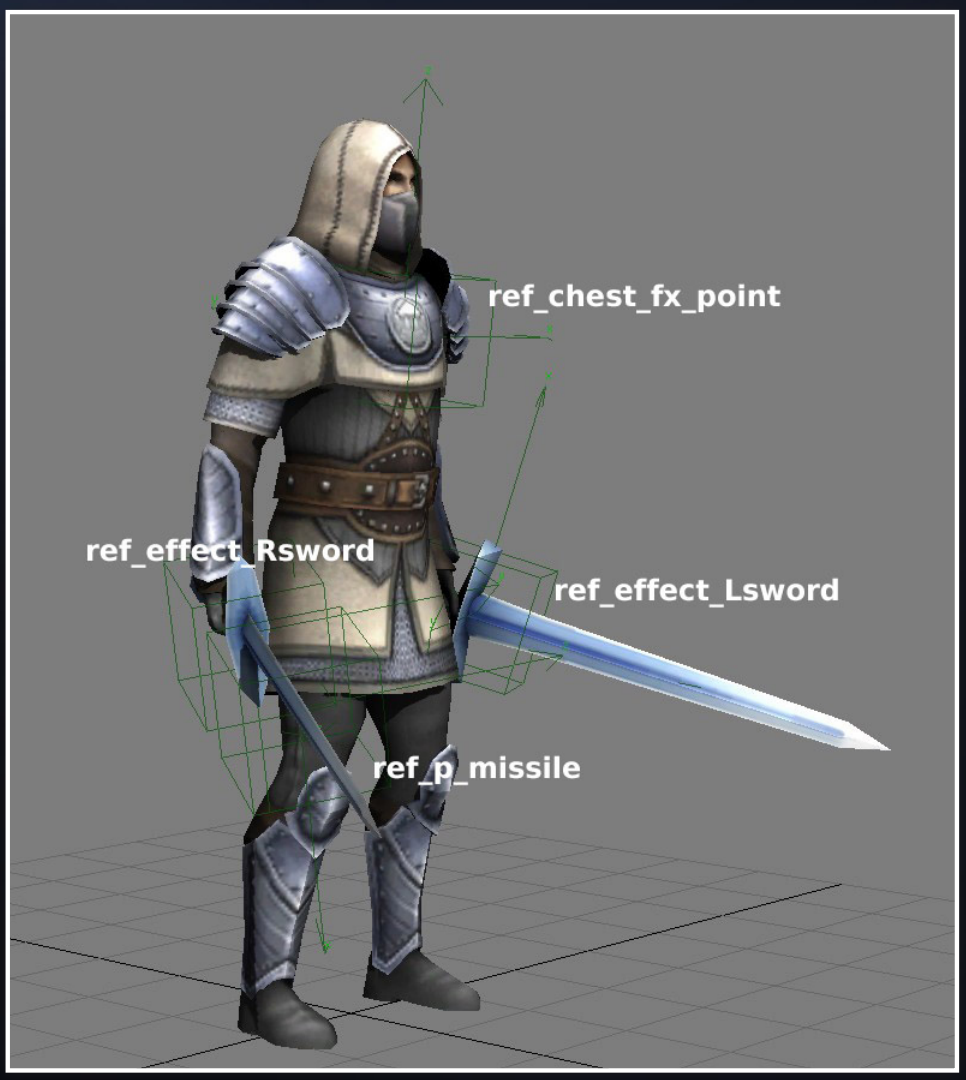
- No more than 4000 polygons.
- Only Skin modification is supported during model export.
- In the zero frame, the model must be set in the Skin-pose at the zero coordinates (origo).
- Animation bones hierarchies for 16 to 200 bones are supported. One vertex of a model can have a maximum of 4 dependent bones. Bones cannot be scaled. There are special tools in the Skyfallen export menu to check whether a model fulfils the requirements:



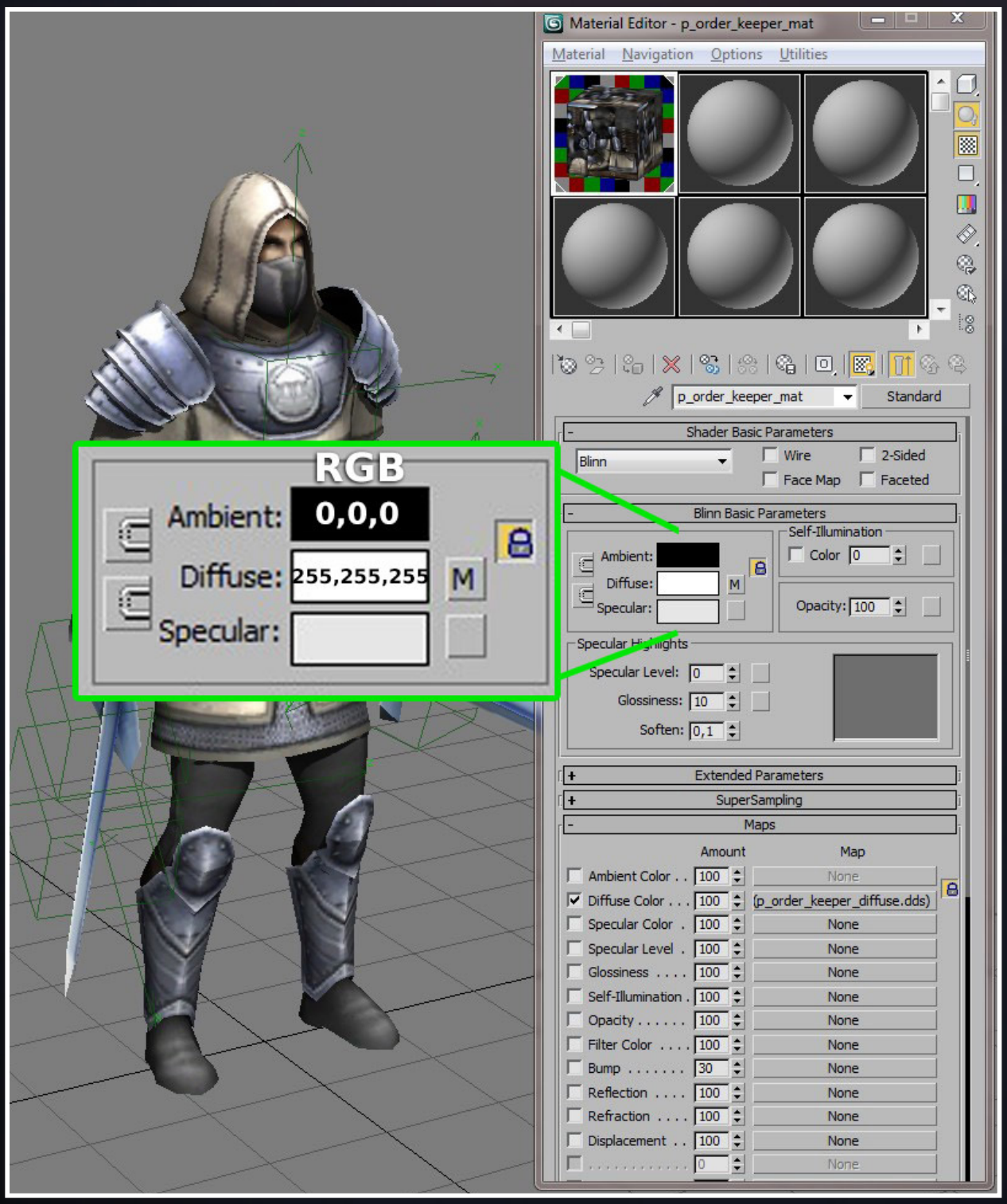
- 512x512 textures are supported. The following formats are

supported: dds dxt1 rgb (no alpha), dds dxt1 argb (1-bit alpha), dds dxt5 argb (8-bit alpha). Additional supported textures: reflection map mask and unit faction colour mask.

➢ To link special effects and logic events to a model, you should use helper points from the Helpers menu. These points should be named as in mask ref_* and the names should be no longer than 32 characters. There are reserved helping points that can be automatically used in the game for the following cases: ref_p_missile* - a point from which a shot or spell effect is drawn; ref_chest_fx_point – a point in which the hit special effect is drawn if an attack hits a unit. All helper points must be linked to the bones or meshes of a model.

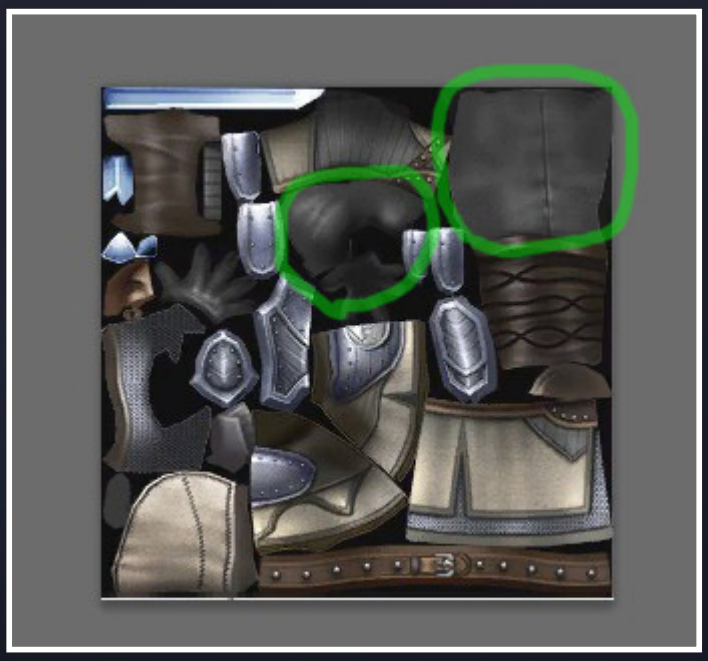


➢ A model has to have the Standard material applied with the corresponding diffuse map and the following parameter settings:

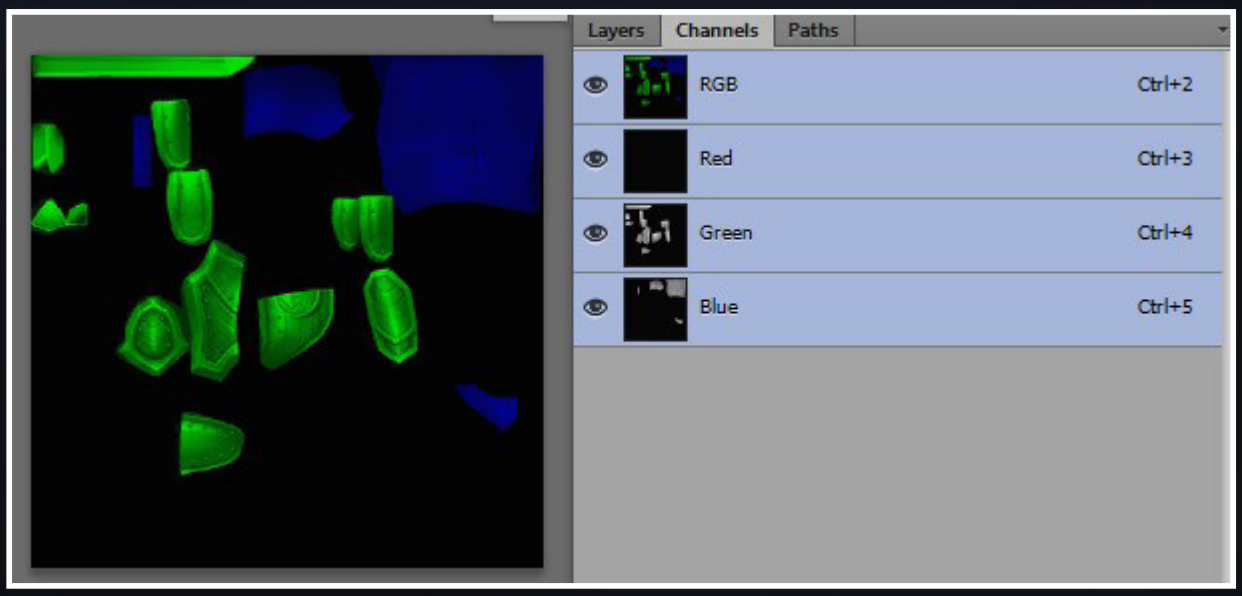

RGB
Ambient: 0,0,0
Diffuse: 255,255,255
M
Specular:
Material Editor - p_order_keeper_mat
Material Navigation Options Utilities
p_order_keeper_mat
Standard
Shader Basic Parameters
Blinn
Wire
2-Sided
Face Map
Faceted
Blinn Basic Parameters
Self-Illumination
Color 0
Ambient:
Diffuse:
Specular:
Opacity: 100
Specular Highlights
Specular Level: 0
Glossiness: 10
Soften: 0,1
Extended Parameters
SuperSampling
Maps
Amount
Map
Ambient Color . . 100 None
Diffuse Color . . . 100 (p_order_keeper_diffuse.dds)
Specular Color . 100 None
Specular Level . 100 None
Glossiness . . . . 100 None
Self-Illumination . 100 None
Opacity . . . . . 100 None
Filter Color . . . . 100 None
Bump . . . . . 30 None
Reflection . . . . 100 None
Refraction . . . . 100 None
Displacement . . 100 None

Texture creation

The diffuse texture of the zones that will appear as the faction colour in the game should be desaturated:



Additional masks should be created in a separate file in RGB channels:



G – reflections mask

B – faction colour mask

You can also add normal maps to the texture.

Exporting a unit with animations

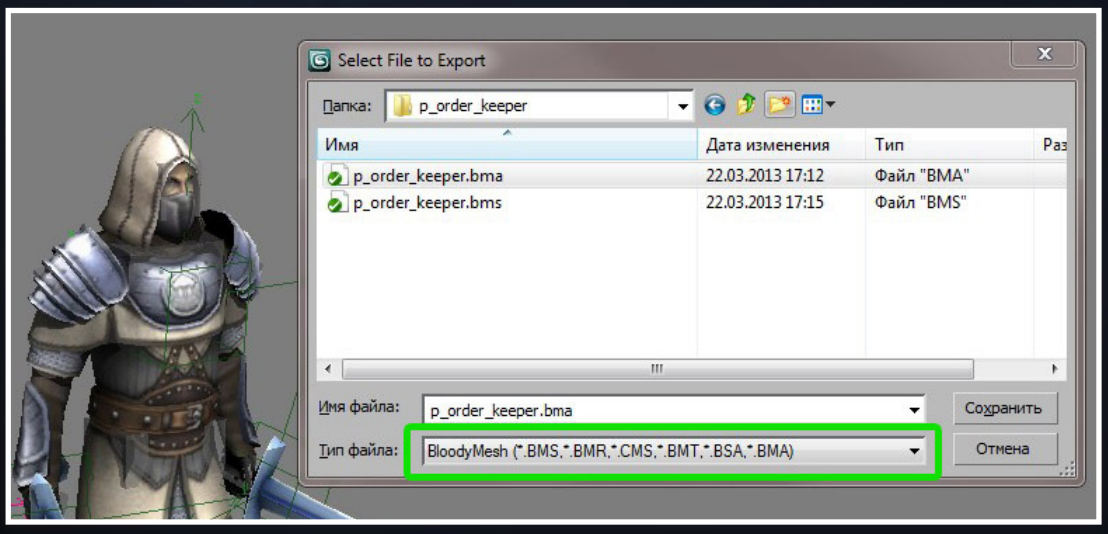
The following animations are supported:

- Idle - idle animation (unit doesn't attack, fight or move).
- idle_random – random idle movement or action (like stretching or checking the weapon).
- attack – main attack animation.
- special - special animation (attack, special action of a unit, etc.)
- death – death animation
- walk – movement animation

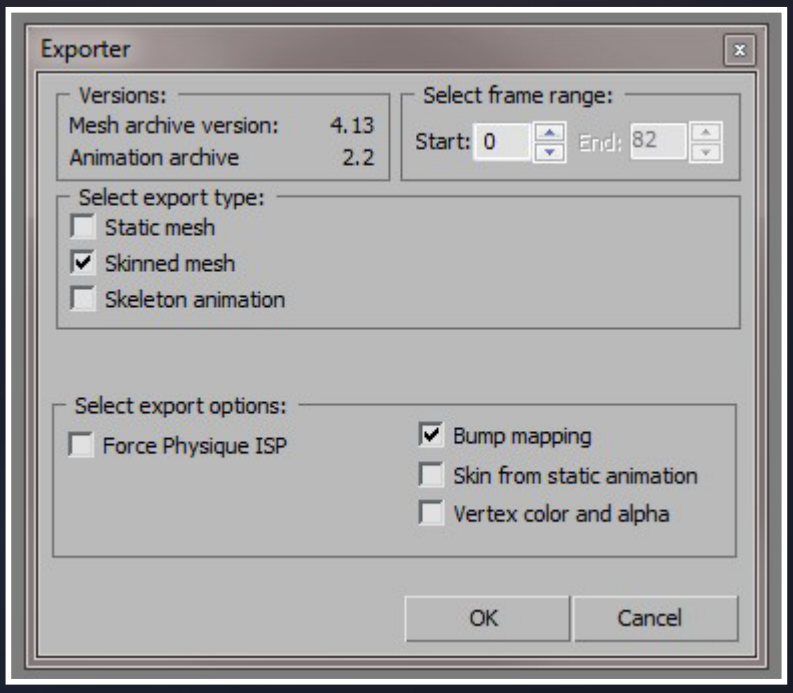
Each of the animations should be saved in a separate file.

How to export a model:

1. Check if the model fulfils all technical requirements. All required scene objects (meshes, animation bones, helper points) need to be visible (not concealed) or the exporter won't 'see' them.
2. Click 'Export' in the SkyFallen menu.
3. In the open window, choose the BloodyMesh file format and name the file:

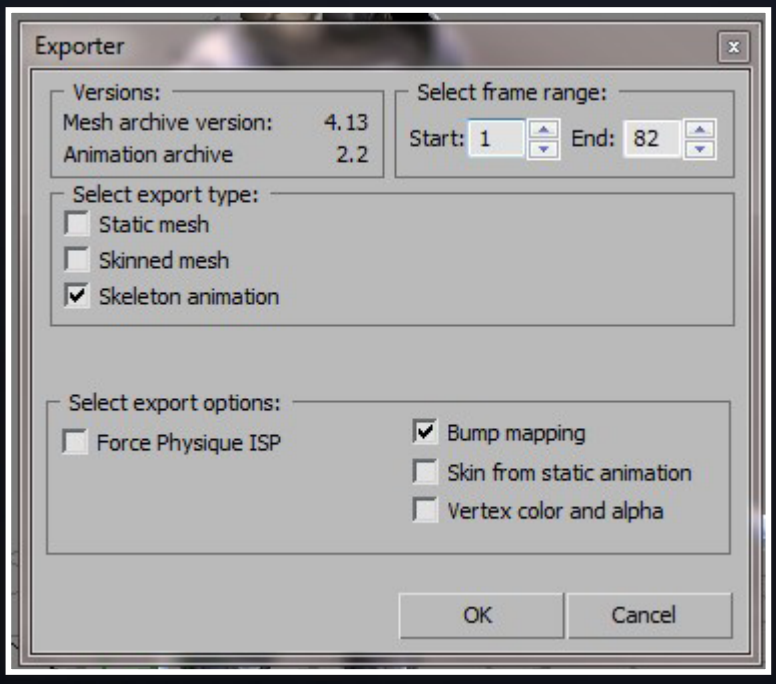


4. Set export parameters as shown below:



5. Press OK. The export will be saved in a file with a .bma extension.

6. To export animations, open the desired animation file and repeat the above process, but this time use the export settings found below:

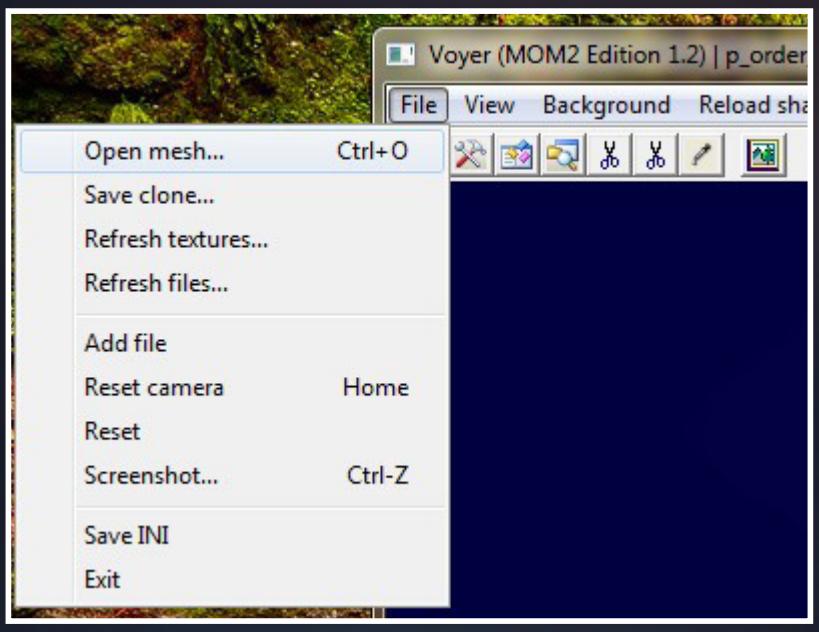


Pay attention to the frame range for every single animation! If there are animated objects that are not linked to skeleton animation, uncheck 'Skin from static animation'. The file will be exported as a .bsa file

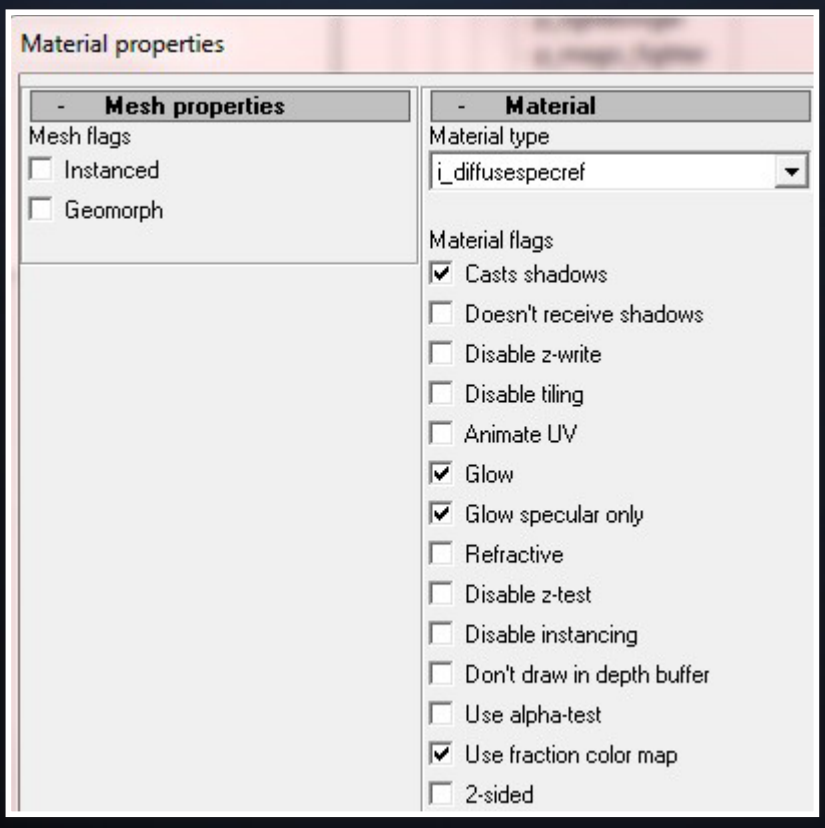
7. The exported files should be put into a Mod folder. All required textures also need to be copied to the Mod folder.

8. Launch voyer.exe (provided along with the game).

9. Open the exported unit's .bma file.



10. Adjust the model's materials. Use the 'Material properties' screen to select the correct type of material:



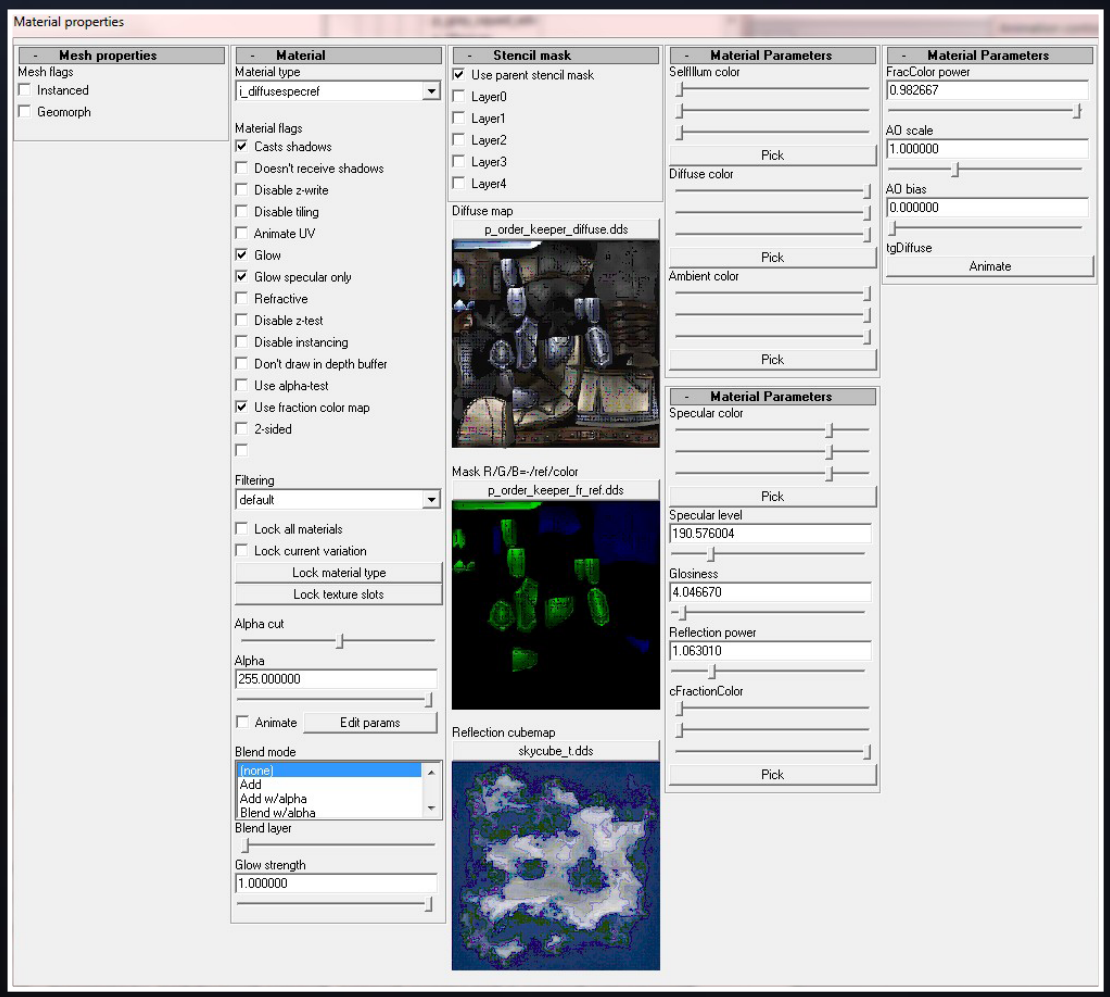
The following types are available:

➛ i_diffuse – This material type only displays the diffuse map.

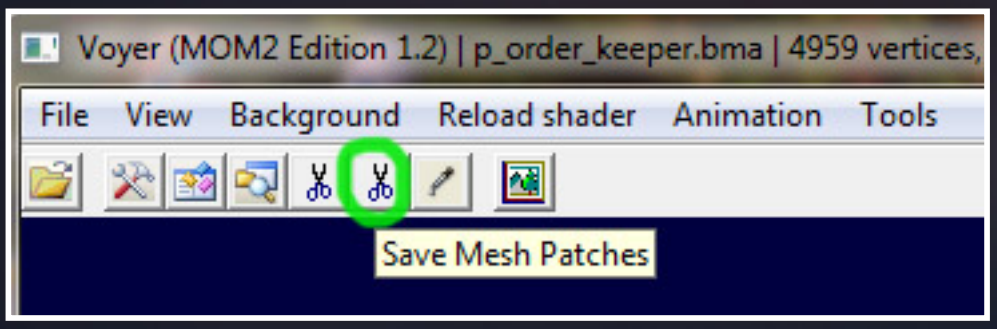
- i_diffuseref – This material type displays both the diffuse and reflection maps.
- i_bump – This material type displays the diffuse map and normal map.
- i_bumpref – This material type displays the diffuse map, the normal map and the reflection map.
- i_diffusespecref – A special material type with a diffuse map and a reflection map. Can be used for imitating metal surfaces with flares and a glare aura.

Next, set up the selected material according to properties desired (should it cast a shadow, should it glow, should it be transparent (using alpha-channel), how should the alpha be applied, should it use the faction colour mask, is it a two-sided material, etc.)

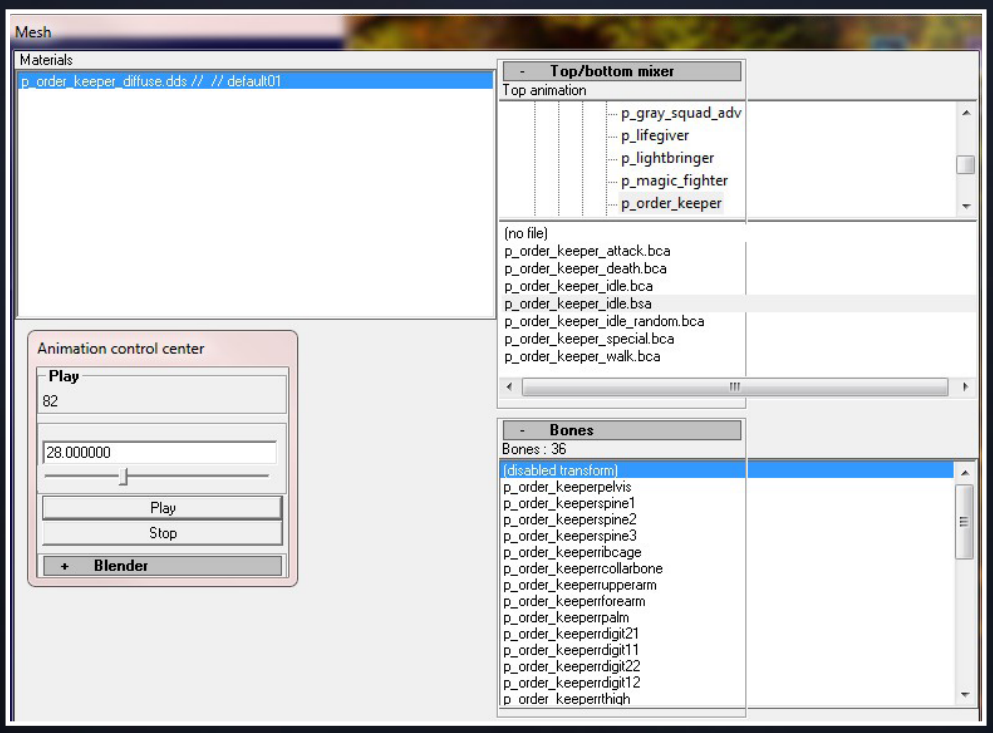
Here is an example of a material after set up:



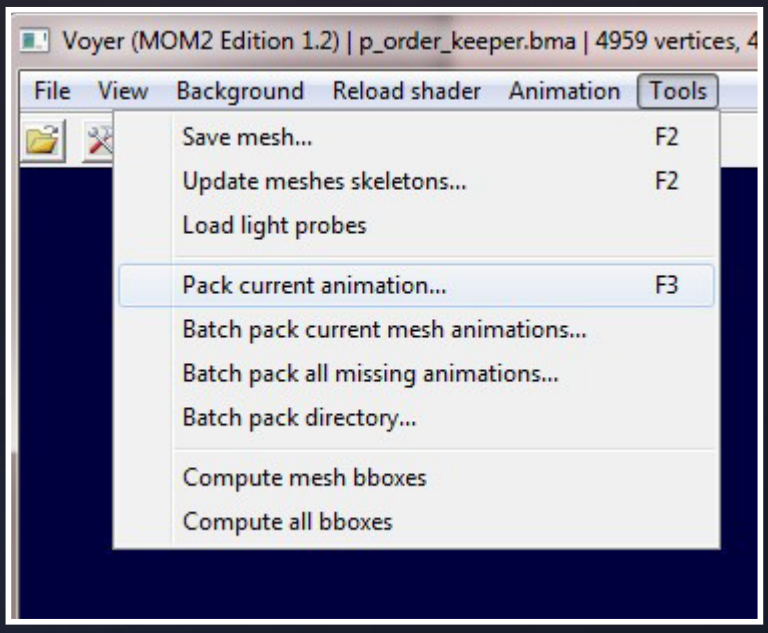
11. Save the material and its settings to the folder with the rest of the model's files:



12. Prepare the animation:
From the Mesh screen, select the animation's .bsa file from the list:



With the help of the Animation Control Centre, you can see the selected animation on the model. If everything looks good, optimize the animation with the special instrument and then pack the animation:



These steps will result in a .bca file that can be used in the game.

Repeat the process for every animation.

Setting a unit's data for the game

To add the graphical assets of a unit into the game, you have to register them in viewData.xml. Use the 'grid' mode in your xml editor. This is the most suitable mode for the task. Find the 'Animations' section:

animations → items → Item (391)

	s_name	s_mesh	i_figureCount	f_movingSpeed	f_rotatingSpeed	f_scale	f_yOffset	f_animBlendTime	b_massiveDeath	b_massiveAttack	s_formationName	s_size
1	p_order_keeper	p_order_keeper.bma	5	12	7	2.95	0	0.5	true	false	Skirmish	SMALL
2	a_cyclop	human_militia.bma	1	16	8	2	0	0.5	true	false	Giants	LARGE
3	a_dragon_gold	a_dragon_gold.bma	1	14	6	2	0	0.5	false	false	Giants	LARGE
4	a_dragon_red	a_dragon_red.bma	1	14	6	1.8	0	0.5	false	false	Giants	LARGE
5	a_dwarf_war	a_dwarf_war.bma	8	10	8	2.1	0	0.5	true	false	Phalanx	SMALL

In the first column, 's_name', enter the system name of a unit from the game data (see the manual on unit creation).

- s_mesh - the name of the .bma file with the model of the unit
- i_figureCount – number of 'soldiers' (separate model copies) in the unit
- f_movingSpeed – visual movement speed of the unit
- f_rotatingSpeed – turning speed of the unit
- f_scale – model scale (1.0 means that the model will be 100% of its size in 3ds Max)
- f_yOffset – vertical offset of the model (for flying units)
- f_animBlendTime – interpolation time between the animations
- s_formationName – 'soldier' formation in the unit
- s_size – scale of the unit (for choosing its banner drawing position)

Then you need to fill in the animation parameter columns:

Idle → file (2)

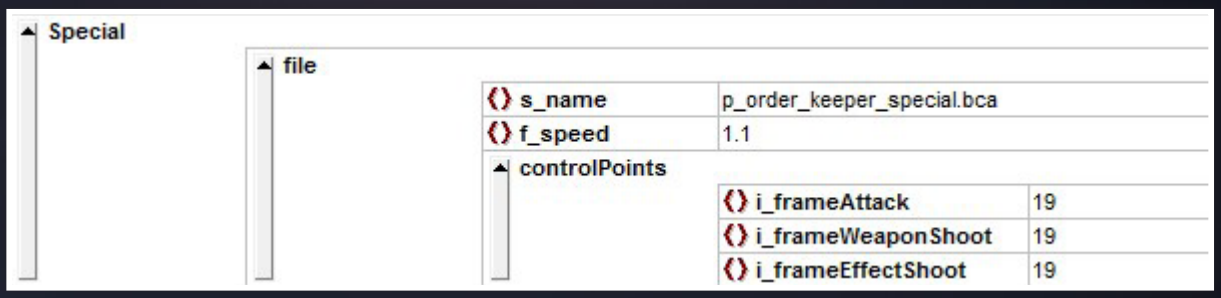
	s_name
1	p_order_keeper_idle.bca
2	p_order_keeper_idle_random.bca

Walk → file (1)

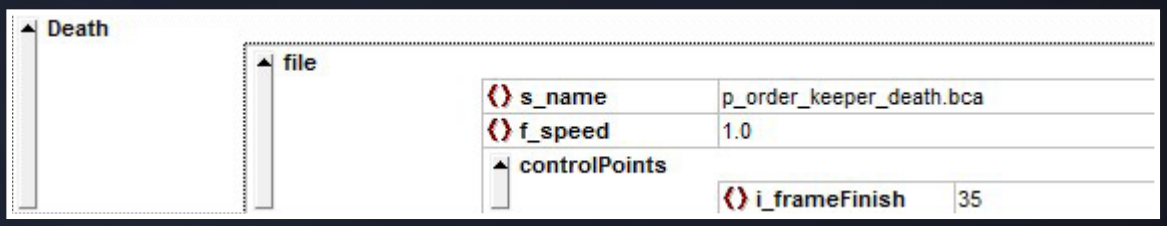
	s_name	f_speed
1	p_order_keeper_walk.bca	1.1

The .bca files for animations should be directed here. You can set the f_speed parameter for each animation. f_speed determines the animation's play speed (1.0 is the animation's default speed from 3ds Max).

For 'attack' and 'special' animations, you should set the 'attack applying' frame. That is the frame the game interprets as a 'moment of action' (when a shot should be fired from a bow, or when a sword hit should cause a special effect to be played, etc.) The i_frameAttack is sufficient for melee attacks. For magical and range attacks you should also add i_frameWeaponShoot and i_frameEffectShoot:

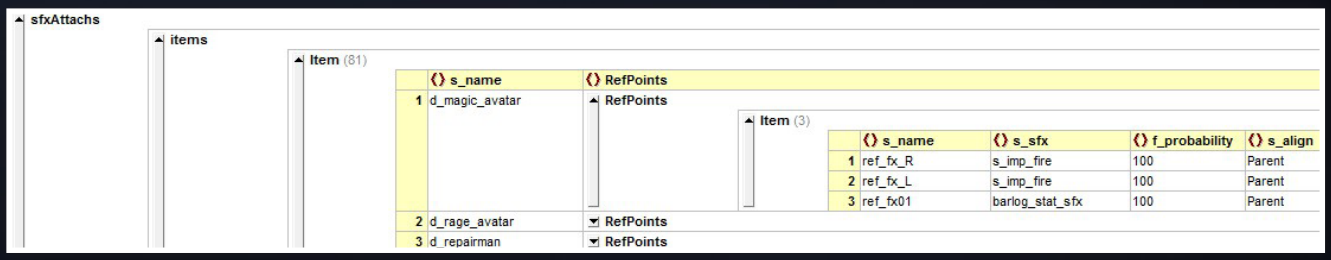


For the death animation, set the frame in which the 'body' should hit the ground:



Linking special effects to a unit

To link special effects, fill in the sfxAttachs section of viewData.xml.



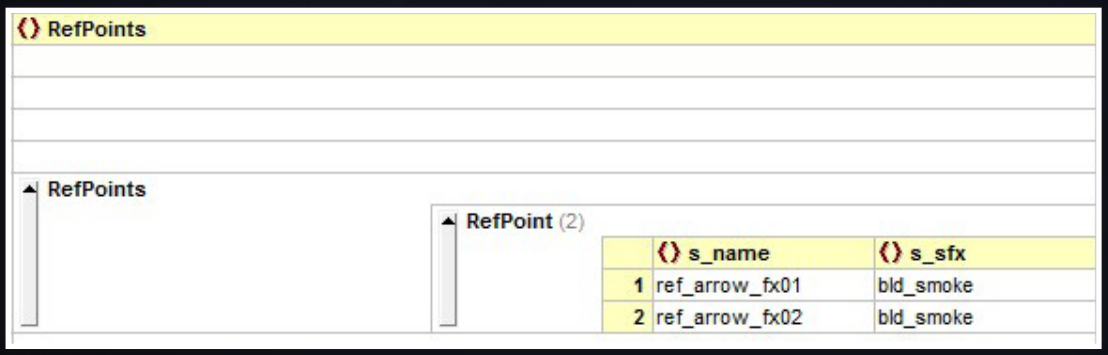
Here you enter the 'system name' of the unit and add the names of the special effects to the corresponding helper points. You may also use the default helper point named 'root' to link an effect to the unit's 'feet' (to the local zero coordinates). Additionally, you can set the probability rate for each effect – how probable it is that the effect will occur.

Setting special effects for a unit's action

To link special effects to unit actions, edit the section 'actionViews' in viewData.xml. In the first column you should set the system name of the action (note that the base attack action must always be named as the unit itself). The other columns are used to edit the visual appearance of the action's effect:

- i_countMissile – number of 'missiles' shot by ships and guard towers.
- i_rangeCountMissile – allows you to set a random number of guard tower missiles using the following formulae:
- Minimum number of missiles = i_countMissile – i_rangeCountMissile
- Maximum number of missiles = i_countMissile + i_rangeCountMissile
- Works only for guard towers!!!

- b_directPointing – missile trajectory switch (true – missiles fly straight into the target, false – missiles fly in an arc).
- b_waterBattle – set 'true' for ships if the ship should turn 90 degrees to port before firing.
- b_specialAnim – set 'true' if you need the action to use a special animation (see **Setting the unit's data for the game**).
- s_shootEffect – name of the special effect that spawns in the initial point of a shot or spell (e.g. flames from a dragon's mouth or smoke from a gun barrel).
- s_shootEffectRefPoint – name of the helper point/points on the model from which the shot or spell is 'coming'. You can use masks if there are several points (e.g. ref_p_missile*).
- s_shootEffectRefPointR – second set of helper points (is used for ships to fire from starboard).
- b_shootEffectFix – set 'true' if you need the special effect from 's_ shootEffect' to be linked to the point of 'shooting'.
- s_hitEffect – name of the special effect that will spawn when the shot/ spell hits its target.
- f_flySpeed – missile flying speed.
- s_flyEffect – name of the special effect on the missile (e.g. flame on 'burning arrows').
- s_flyMesh – the missile's 3D model.
- s_flyMeshScale – scale of the missile's model (1.0 is the actual size of the model in 3ds Max).
- f_flyAnimSpeed – speed of the missile's model animation (e.g. speed of an axe turning during its flight).
- RefPoints – helper points on the missile's model to which the corresponding effects should be linked. Here is an example:



- s_aoeAttEffect – multihex special effect on a ‘friendly’ unit which becomes the centre of the effect (e.g. massive damaging spell around your unit of mages).
- s_aoeDefEffect – multihex special effect on an enemy when the shot/spell hits its target.

We recommend creating special effects for new units by copying the settings from existing ones.

Buildings

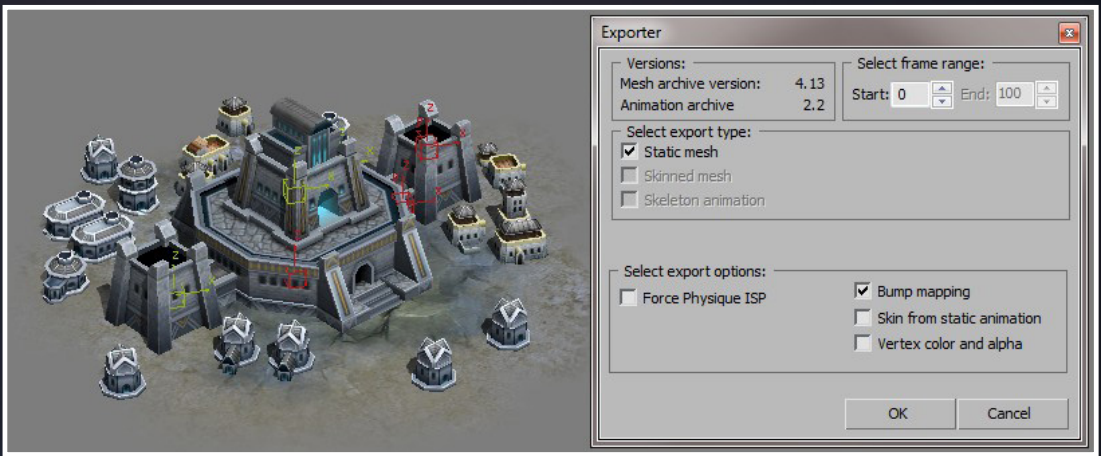
Technical requirements for building models:

- No more than 3000 polygons.
- 512x512 textures are supported. Supported formats: dds dxt1 rgb (no alpha), dds dxt1 argb (1-bit alpha), dds dxt5 argb (8-bit alpha). Supported additional textures: reflection map mask and unit's faction colour mask.
- One cycled animation per building (optional). The name of the animation file should be the same as the name of the building.
- To link special effects and logic events to a model, you should use helper points from the Helpers menu. These points should be named as mask ref_*. The names should be no longer than 32 characters. There are reserved helping points that can be automatically used by the game for the following cases: ref_p_missile* - a point from which a shot or spell effect is drawn, ref_attack_p* – a point where the special effect for a building being hit is shown, ref_flag* - a point at which the flag with the faction colour will appear.
- A model has to have the Standard material (see the requirements for units).
- Textures for buildings must be prepared in the same way as unit textures.
- One special object named Bg_stamp may be added to the scene. Bg_stamp is a flat surface for background landscape texture that will be applied automatically to the landscape under the buildings in the game.

Export

How to export a model: Check whether the model meets all technical requirements.

1. Click 'Export' in the SkyFallen menu.
2. In the open window, choose the path, select the BloodyMesh file format and name the file.
3. Set export parameters:



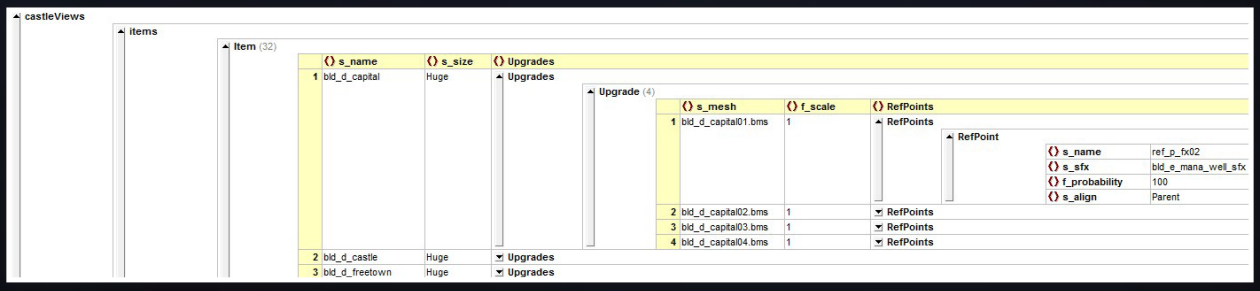
4. Press OK. The exported file with the model will be in .bms format.
5. Complete the next steps as you would with a unit (from point 7 and on).

Setting a building's data for the game

To add the graphical assets of a building to the game, you have to register them in viewData.xml.

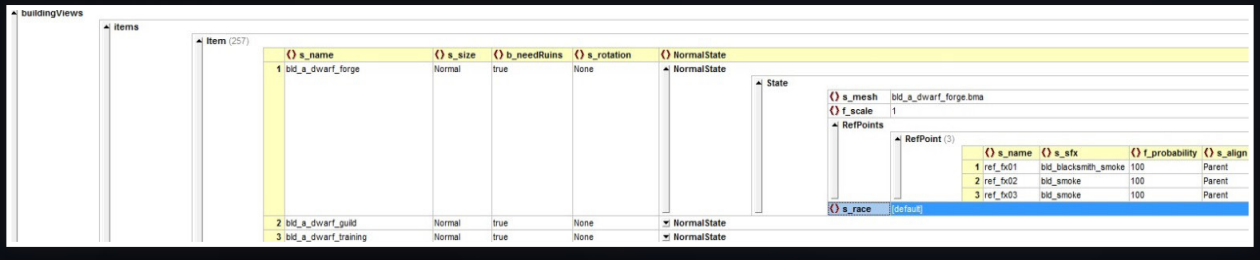
You will need the following sections: castleViews, buildingViews, resourceViews and specialViews.

CastleViews – visual settings for all cities (castles) in the game.



Here the system name of the castle, its logical size (for matching to the right part of the building) and data on its model are entered. Castles have upgrade stages and there are separate models for each stage. For every upgrade model you can independently set the scale and helper points for special effects.

BuildingViews – a section that covers most of the buildings in the game. Here you can edit race buildings, monster lairs, etc. The settings are mostly the same as in the previous section, but there are some additional options:

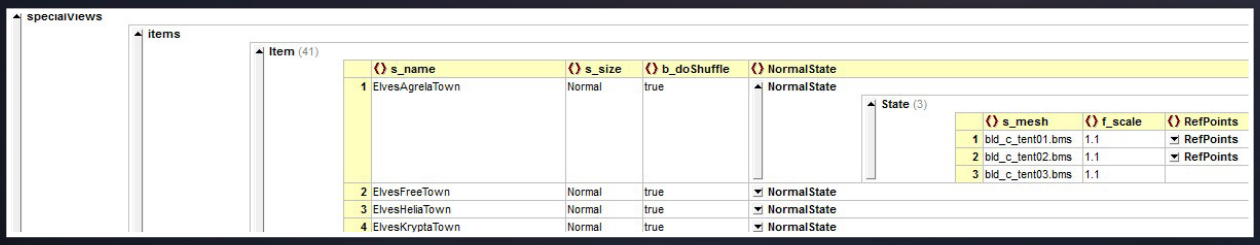


You can decide if the destroyed building should leave debris, if the building should be

oriented toward the water (for shore buildings) and what designates the building's 'race'.

ResourceViews – a section where all models for the local in-game resources are set (local resources are the resources that can be found in the landscape: iron, silver, gold, pumpkins, etc.)

SpecialViews - special buildings like a free cities or citadels. The main building of the special city should be named in the CastleViews group, and here you can pre-set the models for the surrounding hexes:



Important technical details for advanced modding:

Data for all non-published mods is stored in:
Steam\userdata\<userid>\205990\local\mods\

Icons for mods are stored separately in:
Steam\userdata\<userid>\205990\local\icons\

Imported images and temporary 3D model storage for MeshEditor are stored here:
Steam\userdata\<userid>\205990\local\import\

You may create subfolders within those folders using Latin characters.

Published mods to which the player subscribes are stored in pack files.

The mod's description file is stored in the same folder. The pack file or mod folder is: info-file in xml format.

Squeezer.exe should be used to pack and unpack mods:

```
Use:
Squeezer /s archname                show the content of the archive
Squeezer /l archname                extract filenames from the archive
Squeezer /e archname [path]         extract all files to path
Squeezer /f filelist archname [ext1 [ext2 [comp]]] create an archive
using file list
          [ext1] do not compress extensions
          [ext2] do not encrypt extensions
          [comp] use compression level 0..9 (default 1)
Squeezer /d path archname [ext1 [ext2 [comp]]] create an archive using
a tree folder
          [ext1] do not compress extensions
          [ext2] do not encrypt extensions
          [comp] use compression level 0..9 (default 1)
```

Use this line for ext1 and ext2: **.wav.ogg.ogm.mp3.avi.dds.png.tga.jpg.fev.fsb**

All operations with Squeezer should be done from the command line.

You may use the same utility to unpack base game data.

Most mod files are stored in the binary format '.XR'. The format used inside XR is XML.

To unpack and repack .XR files, use TXMLConvert_Final.exe.

How to use this utility from the command line:

```
Syntax: XRconvert.exe [options] input [output]
files:
          input       input file to convert
          output            default: 'input.ext'. output file. set using
input file with automatic extension
options:
          -t:(bin|text|auto)       default: 'auto'. output type
```

If there is no name for the final file, the utility will create an xml or xr file in the folder containing the source file.

You can do the same from the Explorer. Just drag and drop the file onto it and a repacked or unpacked file will be created next to the source file.

You can use any files with only Latin characters in their names. The maximum name length is 256 characters.

Meshes exported from 3d Max should be placed in the 'Import' folder to finalize settings with MeshEditor.

To use finalized models from MeshEditor, place them in the mod folder.

When the user activates several mods at once, mod data may be overwritten, meaning that unit h_rogue from the first mod will be changed to unit h_rogue from the second mod (since they both have same entity name) and so on.

Race trees (adding a unit to a race)

At the moment of writing, building trees can only be affected by one mod at the time! If one mod has unit mod_h_rogue1 added to the human building tree, and a second mod has mod_h_warrior1 added to a different build in the human building tree, then only the latter will show in the game.

How to add a unit into a race (by programming)

Instructions on how to add a new unit into a race:

0. Run the game.
1. Enter Modding->Races->Modify->Choose the Race.
2. Click the 'Building tree' button.
3. Close the game.
4. Copy the mod file named as in m<any_number>_<userid>_data.xr into a different folder.
5. Use command xrconvert m<any_number>_<userid>_data.xr to extract the .xml from the file. XRconverter can be found in the root

game folder.

6. Open m<any_number>_<userid>_data.xml in any xml-editor.
7. Look for the 'buildingTrees' node.

```
<buildingTrees>
   <items>
      <Item>
         <s _ race>humans</s _ race>
...
      </Item>
   </items>
   <disabled/>
</buildingTrees>
```

8. In the 'postUnits' of the needed building (for example bld_c_farm) write the system name of the unit (you can find it in the same file):

```
<postUnits>
   <s _ Item>mod _ h _ rogue2</s _ Item>
</postUnits>
```

9. Add the new node with the new unit and its prerequisites like this:

```
<building>
   <s _ objectType>mod _ h _ rogue2</s _ objectType>
      <s _ itemType>Unit</s _ itemType>
         <prereqs>
            <s _ uniqueness>None</s _ uniqueness>
            <i _ precityUpgrade>0</i _ precityUpgrade>
            <preBuildings>
               <s _ Item>bld _ c _ farm</s _ Item>
            </preBuildings>
            <preResources/>
            <preLandTypes/>
            <preNearLandTypes/>
            <preGods/>
         </prereqs>
         <postUnits/>
      <postBuildings/>
</building>
```

10. Pack .xml back to xr: xrconvert -t:bin m<any_number>_<userid>_ data.xml

11. Return the mod file .xr back to the mod folder.

Extra modding

You can change data other than that displayed in the editor. First, unpack game data and find data.xr and viewdata.xr there. Then use XRConvert to unpack it and find the container of the entity you want to mod.

To change items in those files that are not present in the mod data file (for example, to change god positions on the religion circle), you will need to cut and paste (and change) the container of needed entities (which is in the root of the data tree). So from data.xr you may add containers to m*********_data.xr , and from viewdata.xr to m**********_ viewdata.xr.

An example of god positions on the religion circle:

```
<gods> - container
      <items> - entities list
            <Item> - entity
                  <s _ name>god _ agrela</s _ name>
                  <s _ icon>icon _ god _ agrela</s _ icon>
                  <p2 _ position x='-0.5' y='-0.85'/>
                  <s _ avatar>g _ agrela</s _ avatar>
            </Item>
      </items>
</gods>
```

You may create a new god in the same way.