# Blender NIF Plugin Documentation

*Release 2.6.0a0.dev4*

**Amorilia, neomonkeus**

**Sep 22, 2018**

# Contents

**Warning:** Documentation is incomplete, and is still a work in progress.

The Blender Nif Plugin is a Blender addon that adds support for import and export NIF files. The Nif Plugin allows the user to create custom content for a variety of games which use the nif format.

**Features:**

- Native support for reading & writing of .nif files.
- Integration with Blender modules: Mesh, Texture, Armature, Animation & Collision Systems
- Custom UI to expose Nif specific data
- Utility scripts

**This is one of the many nif-related projects developed by Niftools, our open source community.**

- For more projects see www.niftools.org
- For latest news follow us - @niftools
- For support check out Forums

Contents:

CHAPTER 1

# User Documentation

**Warning:** User documentation is still rather incomplete.

Contents:

## 1.1 Installation

### 1.1.1 Install Blender

Download the recommended version of Blender for your platform (32-bit or 64-bit; if unsure, pick the 32-bit version) and follow the instructions.

- Ensure that the version of Blender being installed is supported by the Blender Nif Plugin.

#### Fedora

For Fedora 16 and up, Blender 2.7x is available via yum:

```
sudo yum install blender
```

### 1.1.2 Install and Register the Addon

1. Download the blender nif plugin .zip release.
2. Start blender.
3. Go to: **File > User Preferences > Add-Ons**.
4. Click **Install Addon…** (bottom).

5. Select the blender nif plugin .zip file downloaded earlier.
6. Under **Categories** (left), select **Import-Export**.
7. Tick the empty box next to **Import-Export: NetImmerse/Gambryo nif format**. You may have to scroll down a bit first.
8. Close the **Blender User Preferences** window.
9. The nif importer and exporter should now show under **File > Import** and **File > Export**.
10. Do: **File > Save User Settings** (unless you enjoy enabling the addon every time when blender starts :-) ).

## 1.2 Workflow

This section will explain the general workflow used to import/export .nif files to/from blender.

- **The nif importer and exporter is show under File > Import** and **File > Export**.

**Note:** This assumes at least begginer level skills with blender. If you need an introduction consider reading the Blender Manual.

### 1.2.1 Import

You now can import the nif directly into blender! These settings are explained at *I/O Settings Section*.

### 1.2.2 Export

The following section deals the various model components, required to export to the .nif format.

| | |
|---|---|
| **Base Model** | Create your base model and configure the settings on the object, refer to the *Geometry Section*. |
| **General Nif Info** | Add in the general info required to identify the .nif file, refer to the Nif Version Info. |
| **Shaders** | Create and configure the shaders for each selected object, refer to the *Shader Section*. |
| **Material** | Create and configure the materials for each selected object, refer to the *Materials Section*. |
| **Textures** | Add the different textures types for each material, refer to the *Textures Section*. |
| **Exporting** | You're now ready to export! These settings are explained at I/O Settings Section. |

### 1.2.3 Advanced Modeling

The following are advanced topics and optional for most models. You should only attempt the following when you are compentent in the basics.

| | |
|---|---|
| **Collisions** | Create collision objects and update their collision settings, refer to the *Collision Section*. |
| **Armatures** | Create a rigged model and configure the flags for each selected bone, refer to the *Armature Section*. |
| **Animations** | Not yet supported. |

## 1.3 User Interface

The Blender Nif Plugin both integrates and expands Blender's UI. This section of the documentation outlines which sections of the default options are used. It also describes any custom UI that is added by the plugin.

### 1.3.1 Import and Export Operators

When the Blender Nif Plugin is enabled via the addon systen it adds new options in the main File menu. The nif importer and exporter will be accessible via the corresponding **File > Import** and **File > Export** menus. Selecting **NetImmerse/Gamebryo(.nif)** option will open the Main UI window.

For a detailed breakdown of all the settings see *io-settings*

### 1.3.2 UI Logging

The Blender Nif Plugin outputs the progress of Import/Export execution via the Information View. There are 3 levels of logging information - Information : This is general progress information - Warning : There was an issue that did not cause the execution to fail, but probably something the user needs to resolve. The full set of warnings will appear at the end in a pop-up window. - Error : There was an issue encountered that caused the execution to fail.

## 1.4 Features

All features of our Blender plugin are described here.

If in need of a guide to make sure your model is complete please visit the *workflow page*.

> **Warning:** Please note that some are still being ported from Blender 2.49b and are marked appropriately.

### 1.4.1 Import and Export Settings

This section explains the import and export settings.

#### Common Settings

There are some common settings shared between the import and export operators.

#### Log Level

- The level at which a log entry is generated to the console window. This is used mainly used for debugging and error checking.
- As a user you will only need to alter this setting if you experience an issue during the import it and a developer asks for more detailed logs than are produced with the default logging level.

> **Warning:** Only a subset of these settings are currently supported even though they have been documented. This is due to the fact that they are ported directly from the old plugin and as such, will functionally remain the same.

### Import Settings

This section explains the import and export settings.

> **Warning:** Only a subset of these settings are currently used or fully supported even though they have been documented here. This is due to the fact that they are ported directly from the old plugin and as such, will functionally remain the same.

### Scale correction

This value is used to globally re-scale the input Nif data to map correctly to Blender's unit of measurement system. The default setting ensures the imported model fits into the view Blender viewport When importing large-scale nif models, such as structures, a user can edit this value so that the nif is easier to work with.

- The Unit of measurement in Blender is the Blender Unit (BU). The default value is 1 BU = 1 meter, but can be remapped to any measurement system.
- The ratio of a Nif Units (NU) to Blender Units (BU) is 1Nu:10Bu, so we need reduce the nif by a factor of 10.
- The Blender Nif Plugin applies a default correction of 0.1

### Override Scene Information

Overrides any existing niftools scene information with the data from the nif that is about to be imported. See :ref:' Scene Settings<user-features-scene>' for information on what settings are available.

### Keyframe File

Keyframe File ( .kf ) is an animation file using key frame markers A more complex animation file introduced with Skyrim ( .hkx ) is a havok based animation file, not supported by the plugin.

### EGM File

EGM files are a vertex morph animation file, most commonly used for facial animations and lip synch.

### Animation

Animation option, when selected, will import the *keyframe* and/or the *EGM* files listed in the file selection entries.

### Align

Align selects how to align the tail position of non parented bones to the root location.

Options are:

- Re-Align Tail Bone - Aligns the tail of non parented bones to a uniform direction.
- Re-Align Tail Bone + roll - Sets a 90 degree roll to the tail after aligning.

#### Process

Determines what parts of the file to import.

Import options include * Everything - This will import geometry, armature, (keyframe, and EGM if set). * Geometry only - Imports geometry and skips all other parts of the file. * Skeleton only - Imports the armature and skips all other parts of the file.

#### Combine Shapes

Select this option if you want all parts of a nif file to be imported as a single multi-part object. This is useful for keeping nifs organized when they contain several parts. As well as allowing for easier exporting of the nif.

#### Combine Vertices

This options will combine vertices that contain the same x,y,z location and normal data into a single vertex. * Select this when vertex ordering is not critical, non animated objects or animated objects that use a skeleton for the animations, but do not contain morph animations. * Do not use this for any object that uses morph type animations.

### Export Settings

This section explains the import and export settings.

> **Warning:** Only a subset of these settings are currently supported even though they have been documented. This is due to the fact that they are ported directly from the old plugin and as such, will functionally remain the same.

#### Scale correction

This value is used to globally re-scale the Blender system of measurement units to the Nif Format units.

- The ratio of a Nif Units (NU) to Blender Units (BU) is 1Bu:0.1Nu. or each NU is about 10x larger than a BU.
- The Blender Nif Plugin applies a default correction of 10
- The default setting ensures the imported model fits into the view Blender viewport.

#### Game

A list of supported games which the plugin will export working nif files.

#### Process

Determines how to export the data in the blend file.

Export options include * All (nif) - Geometry and animation to a single nif. * All (nif, xnif, xkf) - Geometry and animation to a nif. Generates an xnif, and xkf. * Geometry only (nif) - Only geometry to a single nif. * Animation only (kf) - Only animation to a single kf.

#### Smooth Inter-Object Seams

This option combines the normals data for all vertices containing the same xyz location data along an edge and uses the same normal tangent and bi-tangent values for all affected vertices.

#### Use NiBSAnimationNode

NiBSAnimationNode is specific to "The Elder Scrolls - Morrowind" and should only be used when exporting animated items for that game.

#### Flatten Skin

This option does something to the thing, no really it does, but I can't tell you because it's a sekret.

#### Pad & Sort Bones

Adjusts the number of bones in a given partition to match the total number of bones used for the dismember instance.

#### Max Partition Bones

The maximum number of bones that a single dismember partition can use before starting a new partition.

#### Max Vertex Bones

The maximum number of bone weight values that can be applied to a single vertex.

#### Force DDS

Changes the suffix for the texture file path in the nif to use .dds

### 1.4.2 Scene

The follow section outlines Scene settings which is mainly related to Nif Header information. Changing the setting also alters what is displayed by the UI.

The setting can be viewed in the Niftools Scene Panel. This is visible in the Scene Tab of the Properties Editor View.

#### Nif Version

**Nif Version** The base version, generally related to a single game or company. * This is currently not very user friendly, this will be addressed in a future release * Check the nif files included with the game you're modding to know which versions to use.

***Example:*** Nif Version 335544325 is the version that is used for Oblivion.

**User Version** A two digit single integer sub value of Nif Version.

***Example:*** 11 would designate Fallout 3 as the specific game file.

**User Version 2** A second two digit single integer sub value, with the same function as User Version.

***Example:*** 34 would designate Fallout 3 as the specific game file.

---

**Note:**

- Fill in the **Nif Version**, **User Version** and **User Version 2** adequate for your game.
- All three values are used to verify which data should be attached to a file during the export process.
- The scene version is checked at export and compared with the intended export format's version.
- Mismatches will trigger and error and alert the user so that corrections can be effected.

---

### 1.4.3 Geometry

#### Mesh Geometry

- The following section deals with `Object` which are of type 'MESH', containing Mesh Data(Mesh-Object)
- Each `Object` is exported as a combination of `NiTriShape` and `NiNode`.
- The `Mesh` is exported to a `NiTriShape`'s `NiTriShapeData`.

**Example:**

1. Start Blender and empty the scene.
2. Create any Mesh-Object to the scene, eg. cube primitive:
3. Give the Object an appropriate name.

- In the **Object Tab**
- Generic names are automatically generated, unique names helps distingush objects,
- The Nif format only supports triangle based geometry.
- Blender quads and n-gons are exported as triangles, which may lead to differences in rendered geometry.
- Strips (`NiTriStrips`) are available but not developer supported as they are unnecessary for current hardware.

#### Double Sided Mesh

- Each triangle is composed of 3 vertices, edges and a face.
- To decide which way the face is pointing a vector(normal), perpendecular to the face is used.
- This normal vector can be flipped to either side of the triangle; a common source for triangles appearing to not render.

Sometimes you want to allow the mesh to ignore the normal and render both sides, eg a cloak.

- In the **Properties** Editor, in the **Object Data Tab**
- Enable/Disable **Double Sided**, see notes for more detail.

**Double Sided Mesh - Adds a `NiStencilProperty` or similiar,** see *Properties - Stencil Property* for more info.

### UV Unwrapping/Mapping

- UV-unwrapping is the process of unfolding all the faces onto a flat plane, see Cube Unwrap.
- The UV-unwrapping process creates a UV Map layer.
- The UV Map Layer is used to layer connect a `Texture` to `Mesh` and allows visual representation of where each face is located on texture.
- Each `TextureSlot` requires that the user select the UV Map Layer that was generated by unwrapping, See *Textures Section*.

**Example:** #. *Create a mesh-object*. #. In **Edit Mode**, select the faces you want to unwrap. #. Press U‘‘, select **Unwrap > Smart UV Project**.

**Notes:**

- UV-unwrapping adds a `MeshTextureFaceLayer` to the Object.
- Although Blender allows multiple `MeshTextureFaceLayer`, most versions of the Nif format only support one UV layer

### Vertex Color

**Example:**

1. *Create a mesh-object*.
2. Switch to Vertex Paint mode, this automatically adds a base vertex color layer.
3. Apply the desired vertex colors evenly to the vertex.
4. Ensure you have added a *material*.

**Notes:**

- The Nif format only supports a single color per vertex, whereas Blender vertex color per face vertex.
- Blender treats the vertex as if the faces had been split apart, each face can have a different color for that vertex.
- This image should clarify per-face vertes coloring
- On export the scripts will take an average of colors.

### Vertex Alpha

Vertex alpha is handled in the same way as vertex color. The only difference is that vertex alpha use grey scale.

**Example:**

1. *Create a mesh-object*.
2. **Switch to Vertex Paint mode, If there are no vertex color layers this will create a new layer.** you will need to add a second layer manually by clicking the + button in the vertex colors control panel located in the object data menu.
3. **In the brush menu on the left side of the screen, leave the color selector in the center and** use the slider on the right side to change the level of shading with white being fully visible and black being fully transparent.
4. Apply the shading to the vertices just as you would for *Vertex Color*

**Notes:**

- Vertex alpha must use the second vertex color layer, even if there is no color applied in first color layer the default color layer must be in place.

### 1.4.4 Object Settings

This section will take you through the base settings required for your model.

#### BSX

**BSX** is a flagging variable that determines how havok will interact with the object:

- Havok = 1
- Collision = 2
- Is armature (?) = 4
- Enable animation = 8
- Flame nodes = 16
- Editor marker present = 32

***Example:*** To enable havok and collision the value would be 3.

---

**Note:**

- The value is equal to the sum of all items that are enabled.

---

#### BS Inventory Marker

**BS Inv Marker** This sets the x, y, z rotation and zoom level of objects for the in game inventory display in games that support the property.

1. With blender in Object mode, open the BS Inv Marker property window and click "+".

---

**Note:**

- **This should only be applied to the Root object:**
  - For rigged meshed this should be applied to the armature, for non-rigged objects it should be applied to the Mesh object.

---

2. Apply desired values to x,y,z to set the preferred rotation.
   (a) Set view to back view and use rotation to achieve the preferred object orientation.
   (b) Copy the values from the rotation display into the x,y,z lines for BS Inv Marker.
   (c) Delete the decimal and remove any numbers to the right of the fourth digit.
   (d) Press alt + R to reset the object rotation back to 0
3. Apply desired value to zoom
   (a) A value of 1 for zoom is the default, lower values .99 to .01 decrease the item size in the menu.

**Note:**

- Rigged objects that use this value may also use *InvMarker Bones*.

#### UPB

The **UPB** is a group of values contained in a single data string. It's use is unknown.

Niftools uses the following values as default for this item.:

- Mass = 0.000000
- Elasticity = 0.300000
- Friction = 0.300000
- Unyielding = 0
- Simulation_Geometry = 2
- Proxy_Geometry = <None>
- Use_Display_Proxy = 0
- Display_Children = 1
- Disable_Collisions = 0
- Inactive = 0
- Display_Proxy = <None>

#### Settings

First we complete the object panel:

1. The **Nif Root Node** determines the kind of block used at the nif's root. Select one from the dropdown box.
2. The **BS Num UV Set** is ????????. Set it to an appropriate number.
3. The **UPB**'s use is currently unknown. It is recommended you leave it at the default value.
4. Set your **BSX Flags**.
5. The **Consistency Flag** is ????????. Select one from the dropdown box.
6. The **Object Flag** is ???????. Set it to an appropriate number.
7. The **Nif Long Name** is ???????. Set it to an appropriate string.

### 1.4.5 Shader

The following section describes the available shader options and their settings.

There are three different kind of shader nodes:

- *BS Shader PP Lighting Property*
- *BS Lighting Shader Property*
- *BS Effect Shader Property*

### BS Shader PP Lighting Property

This shader node is used for things.

- **Alpha Texture** Alpha Texture requires NiAlphaProperty to enable.
- **Decal Single Pass** Decal.
- **Dynamic Alpha** Dynamic Alpha.
- **Dynamic Decal Single Pass** Dynamic Decal
- **Empty** Unknown.
- **Environment Mapping** Environment Mapping (uses Envmap Scale).
- **External Emittance** External Emittance
- **Eye Environment Mapping** Eye Environment Mapping (does not use envmap light fade or envmap scale).
- **Face Gen** FaceGen.
- **Fire Refraction** Fire Refraction (switches on refraction power/period).
- **Hair** Hair.
- **Local Map Hide Secret** Localmap Hide Secret.
- **Low Detail** Low Detail (seems to use standard diff/norm/spec shader).
- **Multiple Textures** Multiple Textures (base diff/norm become null).
- **Non Projective Shadows** Non-Projective Shadows.
- **Parallax Occlusion** Parallax Occlusion.
- **Parallax Shader Index** Parallax.
- **Refraction** Refraction (switches on refraction power).
- **Remappable Textures** Usually seen with texture animation.
- **Shadow Frustum** Shadow Frustum.
- **Shadow Map** Shadow Map.
- **Single Pass** Single Pass.
- **Skinned** Required For skinned meshes.
- **Specular** Enables specularity.
- **Tree Billboard** Tree Billboard.
- **Unknown 1** Unknown.
- **Unknown 2** Unknown.
- **Unknown 3** Unknown/Crash.
- **Vertex Alpha** Vertex Alpha.
- **Window Environment Mapping** Window Environment Mapping.
- **Z Buffer Test** Z Buffer Test.

### BS Lighting Shader Property

Skyrim PP shader for assigning material/shader/texture.

This shader node is used for things.

- **Cast Shadows** Can cast shadows.
- **Decal** Decal.
- **Dynamic Decal** Dynamic Decal.
- **Enviroment Mapping** Environment mapping (uses Envmap Scale).
- **External Emittance** External Emittance.
- **Eye Environment Mapping** Eye Environment Mapping (Must use the Eye shader and the model must be skinned).
- **Facegen Detail** Use a face detail map in the 4th texture slot.
- **Facegen RGB Tint** Use tintmask for Face.
- **Fire Refraction** Fire Refraction.
- **Greyscale to Palette Alpha** Greyscale to Palette Alpha.
- **Greyscale to Palette Color** Greyscale to Palette Color.
- **Hair Soft Lighting** Keeps from going too bright under lights (hair shader only).
- **Landscape** Unknown.
- **Localmap Hide Secret** Object and anything it is positioned above will not render on local map view.
- **Model Space Normals** Use Model space normals and an external Specular Map.
- **Multiple Textures** Multiple Textures.
- **Non Projective Shadows** Unknown.
- **Own Emit** Provides its own emittance color.
- **Parallax Occlusion** Parallax Occlusion.
- **Parallax** Parallax.
- **Projected UV** Used for decalling.
- **Receive Shadows** Object can recieve shadows.
- **Refraction** Use normal map for refraction effect.
- **Remappable Textures** Remappable Textures.
- **Screendoor Alpha Fade** Screendoor Alpha Fade.
- **Skinned** Required For Skinned Meshes.
- **Soft Effect** Soft Effect.
- **Specular** Enables Specularity.
- **Temp Refraction** Unknown.
- **Use Falloff** Use Falloff value in EffectShaderProperty.
- **Vertex Alpha** Enables using alpha component of vertex colors.
- **Z Buffer Test** ZBuffer Test.

- **Anisotropic Lighting** Anisotropic Lighting.
- **Assume Shadowmask** Assume Shadowmask.
- **Back Lighting** Use Back Lighting Map.
- **Billboard** Billboard.
- **Cloud LOD** Cloud LOD.
- **Double Sided** Double-sided rendering.
- **Effect Lighting** Effect Lighting.
- **Envmap Light Fade** Envmap Light Fade.
- **Fit Slope** Fit Slope.
- **Glow Map** Use Glow Map in the third texture slot.
- **HD LOD Objects** HD LOD Objects.
- **Hide On Local Map** Similar to hide secret.
- **LOD Landscape** LOD Landscape.
- **LOD Objects** LOD Objects.
- **Multi Index Snow** Multi Index Snow.
- **Multi Layer Parallax** Use Multilayer (inner-layer) Map.
- **No Fade** No Fade.
- **No LOD Land Blend** No LOD Land Blend.
- **No Transparency Multisampling** No Transparency Multisampling.
- **Packed Tangent** Packed Tangent.
- **Premult Alpha** Has Premultiplied Alpha.
- **Rim Lighting** Use Rim Lighting Map.
- **Soft Lighting** Use Soft Lighting Map.
- **Tree Anim** Enables Vertex Animation, Flutter Animation.
- **Uniform Scale** Uniform Scale.
- **Unused01** Unused.
- **Unused02** Unused.
- **Vertex Colors** Has Vertex Colors.
- **Vertex Lighting** Vertex Lighting.
- **Weapon Blood** Used for blood decals on weapons.
- **Wireframe** Wireframe.
- **Z Buffer Write** Enables writing to the Z-Buffer.

### BS Effect Shader Property

Skyrim non-PP shader model, used primarily for transparency effects, often as decal.

- **Cast Shadows** Can cast shadows.
- **Decal** Decal.
- **Dynamic Decal** Dynamic Decal.
- **Enviroment Mapping** Environment mapping (uses Envmap Scale).
- **External Emittance** External Emittance.
- **Eye Environment Mapping** Eye Environment Mapping (Must use the Eye shader and the model must be skinned).
- **Facegen Detail** Use a face detail map in the 4th texture slot.
- **Facegen RGB Tint** Use tintmask for Face.
- **Fire Refraction** Fire Refraction.
- **Greyscale to Palette Alpha** Greyscale to Palette Alpha.
- **Greyscale to Palette Color** Greyscale to Palette Color.
- **Hair Soft Lighting** Keeps from going too bright under lights (hair shader only).
- **Landscape** Unknown.
- **Localmap Hide Secret** Object and anything it is positioned above will not render on local map view.
- **Model Space Normals** Use Model space normals and an external Specular Map.
- **Multiple Textures** Multiple Textures.
- **Non Projective Shadows** Unknown.
- **Own Emit** Provides its own emittance color.
- **Parallax Occlusion** Parallax Occlusion.
- **Parallax** Parallax.
- **Projected UV** Used for decalling.
- **Receive Shadows** Object can recieve shadows.
- **Refraction** Use normal map for refraction effect.
- **Remappable Textures** Remappable Textures.
- **Screendoor Alpha Fade** Screendoor Alpha Fade.
- **Skinned** Required For Skinned Meshes.
- **Soft Effect** Soft Effect.
- **Specular** Enables Specularity.
- **Temp Refraction** Unknown.
- **Use Falloff** Use Falloff value in EffectShaderProperty.
- **Vertex Alpha** Enables using alpha component of vertex colors.
- **Z Buffer Test** ZBuffer Test.
- **Anisotropic Lighting** Anisotropic Lighting.

- **Assume Shadowmask** Assume Shadowmask.
- **Back Lighting** Use Back Lighting Map.
- **Billboard** Billboard.
- **Cloud LOD** Cloud LOD.
- **Double Sided** Double-sided rendering.
- **Effect Lighting** Effect Lighting.
- **Envmap Light Fade** Envmap Light Fade.
- **Fit Slope** Fit Slope.
- **Glow Map** Use Glow Map in the third texture slot.
- **HD LOD Objects** HD LOD Objects.
- **Hide On Local Map** Similar to hide secret.
- **LOD Landscape** LOD Landscape.
- **LOD Objects** LOD Objects.
- **Multi Index Snow** Multi Index Snow.
- **Multi Layer Parallax** Use Multilayer (inner-layer) Map.
- **No Fade** No Fade.
- **No LOD Land Blend** No LOD Land Blend.
- **No Transparency Multisampling** No Transparency Multisampling.
- **Packed Tangent** Packed Tangent.
- **Premult Alpha** Has Premultiplied Alpha.
- **Rim Lighting** Use Rim Lighting Map.
- **Soft Lighting** Use Soft Lighting Map.
- **Tree Anim** Enables Vertex Animation, Flutter Animation.
- **Uniform Scale** Uniform Scale.
- **Unused01** Unused.
- **Unused02** Unused.
- **Vertex Colors** Has Vertex Colors.
- **Vertex Lighting** Vertex Lighting.
- **Weapon Blood** Used for blood decals on weapons.
- **Wireframe** Wireframe.
- **Z Buffer Write** Enables writing to the Z-Buffer.

### 1.4.6 Properties

The following section describes how Blender object properties/settings map to their Nif counterpart.

### Mapping

- *Read NiProperty*. to see what blender settings are mapped for Nif versions using NiProperty blocks.
- Nif versions which is BSLightingShaderProperty

### Material Based Properties

- The following section goes through those Blender material settings and how they relate to corresponding nif blocks or attributes.
- Unless otherwise stated your mesh needs to have a material.
- The nif format only supports a single material per `NiTriShape`.
- A Mesh which contains mult-materials will be exported as multiple `NiTriShape`.
- Giving the Material an appropriate name helps distingush materials and encourages reuse, eg. Metal, Glass, plastic etc.

### Example

1. *Create a mesh-object* as explained before.
2. In the **Properties** panel, in the *Material* tab click **New** to create a new material.
3. Assign the Material an appropriate name.

- *See material settings* to see what material settings we use.

### Blender Materials Settings

The following section describes which Blender Material setting we actively use. Depending on the nif version you are exporting to, they will be mapped to different Nif block types or block attributes.

### Ambient

This is a global scene value; even if you use several materials they all share the same value.

1. In the **World Tab -> Ambient Color**.

- The diffuse color dynamically calculated, so the value is not actually used.
- If you have found a nif that actually uses these values please contact the devs and we can enable per material ambient.

### Diffuse

1. In the **Diffuse** panel, click on the color to bring up the color widget

- Blender defaults is 0.800, which is off-white.
- See notes above why this value is defaulted on export.

#### Emissive

This value sets how much light the material emit.

1. In the **Shading** panel is the Emissive color
2. Set the **Intensity** value,

- Blender uses the diffuse color for emission, viewable in the viewport.

#### Gloss

This value sets how diffuse the specular highlight across the material.

1. In the **Specular** panel
2. Set the **Hardness**

- This value is used to set how intense the specular highlight should be.

#### Specular

The Specular value create the bright highlights that one would see on a glossy surface.

1. In the **Specular** panel, use the color widget to set the highlight color.
2. Set **Intensity** to whatever value you prefer.

#### Alpha

The alpha component of the material is how much you can see through the material.

1. In the **Transparency** panel, **Enable Transparency**
2. Ensure **Z Transparency** is selected, which is by default.
3. Alter the **Alpha** setting.

#### NiProperty

The following is a overview of what the Nif Plugin will export Blender settings are mapped to.

#### NiMaterialProperty

The following section is for nifs which use `NiMaterialProperty`. * Every `Material` is exported to a `NiMaterialProperty`.

#### NiSpecularProperty

- Setting the **Intensity** to **0** will disable specularity; a `NiSpecularProperty` will not be exported.

#### NiAlphaProperty

- An `NiAlphaProperty` is exported for Materials or Texture have Alpha value.

#### NiWireFrameProperty

`NiWireframeProperty`

#### NiStencilProperty

The NiStencilProperty ignores the face normal and renders both sides of the mesh.

1. In the **Object Tab -> Double-Sided**, enable/disable.

- This will export a `NiStencilProperty`

### 1.4.7 Textures

A texture refers to a generic image.
A map refer to an image which has specific properties which will influence the renderer.
Textures/Maps allow us to alter how the geometry is rendered, such as adding additional detail, affecting light, etc.

#### Notes

- The nif format only supports UV mapped textures, so only those will be exported.
- GLSL mode is enabled on import/export, but should be enabled manually otherwise to give correct viewport preview.
- Relative paths for textures are often used, eg. /Texture/../.. which should be adjusted so Blender can render in the viewport.

#### Requirements

Adding textures requires the following:

1. *A Mesh-object*.
2. The Mesh-object needs to be *uv-unwrapped*.
3. That Mesh-object requires *a Material*.

#### Creating a Texture Slot

Create a Texture slot to hold a texture.

- In the **Texture tab**, Click **New** to create **Texture Slot**.

A texture slot has a default texture, we set type of image we will use:

- Under **Type**, select **Image or Movie**or **Enviromental**, see *texture maps*

We load an image to use as our texture.

- Next to **Image**, click **Open**, and select the desired texture image.

Set the texture to use the UV coordinates.

- Under **Mapping > Coordinates**, select **UV**.

The UV layer that was *create previously* needs to be selected for this texture.

- In the **Texture** tab, under **Mapping > Layer**, click on the empty field, and select `UVTex`.

### Texture Maps

Each Texture Map affects different properties of how the geometry is rendered

### Texture Maps

- Each Texture Map has a distinct function, which affectes how the object is rendered
- The following sections describe the function of each Map
- Map properies are usually set via the **Influence** section of the **Texture Tab**.

### Diffuse Map

The information in a diffuse map is used as the base color

1. Under **Type**, set the type to **Image or Movie**.
2. Under **Influence**, in the **Diffuse Section**, enable **Color**.

### Bump Map

The information in a Bump map is used as affect the lighting of the object to give apparent detail.
This map is usually a grey-scale texture.

1. Under **Type**, set the type to **Image or Movie**.
2. Under **Influence**, in the **Geometry Section**, enable **Normal**.

### Glow Map

A texture that receives no lighting, but the pixels are shown at full intensity.

1. Under **Type**, set the type to **Image or Movie**.
2. Under **Influence**, in the **Shading Section**, enable **Emit**.

### Normal Map

A Normal Map is usually used to fake high-res geometry detail when it's mapped onto a low-res mesh.
The pixels of the normal map each store a normal, a vector that describes the surface slope of the original high-res mesh at that point.
The red, green, and blue channels of the normal map are used to control the direction of each pixel's normal.

**Warning:** Currently not supported.

1. Under **Type**, set the type to **Image or Movie**.
2. Under **Influence**, in the **Geometry Section**, enable **Normal**.
3. Under **Image Sampling**, enable **Normal**

### 1.4.8 Collision Systems

The following section deals with the various collision systems that the Blender Nif Plugins supports.

#### Collision Object

**Warning:**

- This section has not been ported yet, meaning it does not currently work.

The following section deals with creating a mesh-object which will physically represent our collision object. Once a suitable object has been created, then the appropriate settings should be enabled on

#### Bounding Box

This is used as the bound box.

1. Create a Mesh-Object to represent the bound box, a Cube is recommended.
2. Name the object BSBound or BoundingBox, depending on which version you wish to be exported.
3. In the Object Tab, enable bounds in the display section.

#### Havok Collision

This is used by the havok system for collision detection.

**Warning:**

- For Cylinder Export, we need to fix them to show how the user would create the objects. We are using a Meta Capsule
- Some of the collision types lack viewport rendering, see workaround for visulisations below.

#### Notes

- Collision Bounds are represented by a dashed line, unlike Bounds which are by solid lines.

### Collision Object

- The following section describes the most appropriate primitive object to represent the desired collision object type.
- The suggested shapes are the same objects generated through import by the plugin.
- Upon export, a BhkShape is created from data pulled from the Collision object.
- Start by choosing a shape adequate to your model and follow the steps below the appropriate section.
- Oblivion, Fallout 3, and Fallout NV;

| Blender | Nif |
|---|---|
| *Box Collision* | `bhkBoxShape` |
| *Sphere Collision* | `bhkSphereShape` |
| *Capsule Collision* | `bhkCapsuleShape` |
| *Convex Hull Collision* | `bhkConvexVerticesShape` |
| *Triangle Mesh Collision* | `bhkMoppBvTreeShape` |

- Morrowind:

| Blender | Nif |
|---|---|
| *Triangle Mesh Collision* | RootCollisionNode |

### Box Collision

1. *Create your mesh-object*.
2. Create a second mesh-object to represent our collision object, a primitive cube(prim-cube) is recommended.
3. Rename the prim-cube via the Object panel, eg. 'CollisionBox'
4. Scale the 'CollisionBox' uniformly to the size wanted.
5. *Add physics to our 'CollisionBox'*.

### Sphere Collision

1. *Create your mesh-object*.
2. Create another mesh-object to represent our collision shape, a primitive sphere(prim-sphere) is highly recommended.
3. Rename the prim-sphere, eg. 'CollisionSphere', via the Object panel
4. Scale the 'CollisionSphere' object as needed, ensuring all vertices are enclosed by the sphere
5. *Add physics to our 'CollisionSphere'*.

### Capsule Collision

1. *Create your mesh-object*.
2. Create a second mesh-object to represent our collision object, a primitive cylinder(prim-cylinder) is recommended.

3. Rename the prim-cylinder via the Object panel, eg. 'CollisionCapsule'.
4. Scale the collision cube 'CollisionBox' to the size wanted.
5. *Add physics to our 'CollisionCapsule'*.

**Notes:**

- If the lenght is less than or equal to the radius, then a `bhkSphereShape` is generated instead.
- Currently Capsule bounds lack viewport preview, awaiting Bullet Physic integration
- The following is a workaround; **Prone to user error, Ensure to reset setting after visualising!**.
- In the **Object Tab**, under the **Display** section enable **Bounds**
- Set the **Type** to **Cylinder**.
- This shape best represents the capsule, but visually missing the end caps which are hemi-spheres.

### Convex Hull Collision

1. *Create your mesh-object*.
2. Create a convex mesh. See *Notes*
3. Rename the hulled-object, eg. 'CollisionHull' via the Object panel.
4. Scale the collision cube 'CollisionBox' to the size wanted.
5. *Add physics to our collision cube 'CollisionBox'*.

**Notes:**

- It is advisable to use a convex hull generator to create the collision-mesh.

### Triangle Mesh Collision

1. *Create your mesh-object*.
2. Create a convex hulled mesh-object. See *Notes*
3. Rename the polyhedron-mesh, eg. 'CollisionPolyhedron' via the Object panel.
4. Scale the collision cube 'CollisionPoly' to the size wanted.
5. *Add physics to our collision cube 'CollisionBox'*.

**Notes:**

- Often a duplicate object can be used, simplified by decimating, then triangulated(**Ctrl + T**).
- A *Convex Hulled Object* can also be used.

### Rigid Body

1. Go to the **Physics** tab in the **Properties** area.
2. Click on **Rigid Body** to enable this modifier.
   (a) Set a mass adequate for your model.

#### Collision Modifier

#### Collision Settings

The following section details the setting which need to be applied to the collision body to react appropriately in the collision simulation. A lot of these setting are havok specific; where possible we have mapped them to blender's collision simulation.

#### Enabling Collisions

First we enable Collision Setting for the selected Collision Object:

- In the the **Physics** tab, enable **Collision Bounds**
- Meshes with Collision Bounds enabled will be exported as a `bhkShape`, rather than a `NiTriShape`.

The bounds type is used to select which BhkShape type to use.

- Select the desired **Bounds** type from the dropdown box.

#### Havok Settings

- The Collision settings are used by the `bhkShape` to control it reacts in the Havok physics simulation.

#### Worflow

Here it is explained how to add collision to your mesh.

1. Go to *Collision Object* and follow the steps indicated there:
   (a) Choose between a *Bounding Box* or *Havok Collision*.
   (b) The bounding box is complete. If you have chosen Havok Collision proceed to choosing an *appropriate shape* for your collision.
   (c) Add a *Rigid Body* modifier.
2. Go to *Collision Settings* and follow the steps indicated there:
   (a) Start by *enabling collision bounds*.
   (b) Define the the *Havok Settings* for your collision bounds.

### 1.4.9 Armature

#### Armatures

#### Armature Bones

- The following section deals with Armatures

### Bone Flags

- These are set upon export if left at 0.
- Only change the value in very specific cases: Oblivion Clothing uses 0x000F

### Dismember Flags

- Currently does nothing - in testing.

### Inventory Marker

- This is a special type of bone which is used to position object in the inventory display.
- It may also be used for animation placement involving multiple NPCs
- The InvMarker bone should only be used in engines that can support them.

**Example:**

1. Create this item in the same manner as you would for a standard armature bone.
2. Parent must be Armature root.
3. **Naming must start with InvMarker and can only be appended with .000** A model with 4 inventory marker bone items should be named as InvMarker, InvMarker.001, InvMarker.002, InvMarker.003

**Notes:**

- Games known to support this include: The Elder Scrolls - Skyrim.
- Exporting this type of bone into engines that do not support it will cause crashes.

## 1.5 Support

For more information and further support, please contact us at:

- Wiki
- Forum
- Report issues via Github Issues

CHAPTER 2

# Developer Documentation

Contents:

## 2.1 Developer Design Documentation

This section provides a high level overview of the design of the Blender Nif Plugin. It is geared towards anyone interested in delving into developing the Blender Nif Plug-in or knowing more about the development process.

Contents:

### 2.1.1 Development Overview

This section outlines various priciples which govern how we develop the Blender Nif Plugin and why.

#### Development Methodology

A development methodology describes the principles we want as part of the development process.

For the 2.6.x series of development we decided to develop a Feature Oriented, Test Driven Development (TDD) methodology to suit both current and future needs.

**Test driven development in a nutshell means:**

- We write a test for some functionality we want.
- We run the test, initially it will fail.
- We code the feature.
- We re-run our tests, changing the code until it passes.

**The advantages of TDD is:**

- It gives us quick feedback when changes arises.

- Constantly testing the code to ensure that it is doing what is meant to do, no assumptions.
- When things break, it narrows the search; allowing fixes to be developed more quickly.
- Ensure that the changes haven't broken any other existing functionality.

**The focus of our TDD methodology is 3 main areas:**

- Develop maintainable code - Ensure good foundation for stable development & improvement.
- Constant feedback - Testing features ensuring they work and remain working.
- Documentation - Ensure that people can actually use the tools, think about user perspective.

### Code Development & Maintenance

The initial proposal was to port all the current code directly to the new Blender Python API. See *Code Porting Strategy*

Additionally

- It was decided that as code was ported that the conventions as described in the next section would be introduced to keep the code consistent and improve readability.
- Refactored of code into modular components when working on features sets; collisions, texture, armature etc.

### Modularisation

During the code porting process it became apparent that the code was monolithic. All of the import code was in one class, all export code in another. Initially we planned to hold off large scale refactoring until the code was ported and do it as part of the 3.0.x series. It was decided to separate out common areas of functionality into submodules which would be responsible for that specific areas.

- Some systems are still highly coupled, such as geometry generation with the material system, these will remain in place.

Modularisation of code makes it much easier to add on new functionality, such as new collision type, when the code is localised This will also make the refactoring process easier as we can target specific areas.

### Git Development Model

We use git as our version control system. In order to facilitate parallel development of a single code base, we developed a git workflow based on the popular nvie gitflow model.

The goals were:

1. Central clean repository which everyone forks from, avoids new developer cleaning up unwanted branches
2. Developers create feature branch off develop, rebase if develop is updated
3. Central repository to accept pull requests, for peer review.
4. Merge code into develop, developers synch with central

When a develop forks from the central repo, their repo will have only 2 branches, master & develop.

- Their primary use is tracking updates from central; only changes should be those pulled from the central repo.
- Develop is used as the base to create feature branches.
- If develop is updated, then all feature branches should be rebased. This reduces conflicts for pull requests sent to the central repo.

**image** http://i211.photobucket.com/albums/bb189/NifTools/Blender/documentation/Git%20Development%20Model/git_developer_model_zps55d02850.png

When a developer feels that their feature branch is ready they can start the review process

- A pull request needs to be sent to the central repo against the develop branch.
- A review for that specific project will review the pull request
- If they are happy with the changes it will be merged into develop and all active developers will be notified to pull the changes into their local repos.
- If additional changes are required then the pull request if left open and the developer can add commits to their repo which get automatically added to the pull request.

  **image** http://i211.photobucket.com/albums/bb189/NifTools/Blender/documentation/Git%20Development%20Model/git_developer_model_zps55d02850.png

#### Test-Framework

In Test Driven Development, tests are the core to ensuring software quality. Before any production code is written, a test should be written to check to see that the code does what it does. Initially the tests will fail. As the code is developed, then more tests should pass until all tests do. At this point a feature is deemed to be implemented.

Some points of note:

- It is up to the developer to create tests which are appropriate in the level of testing.

Previously, tests were created ad-hoc, based often on bug fixes, so did not extensively test the code. It was decided that we would develop a test-framework to standardise testing in parallel to the porting process. The current goal of the test-framework is to provide integration level testing to ensure features function as required.

The Test Framework has several purposes

- Standardise testing through the use of the template pattern.
- Provide functionality testing as features are ported to the newer api.
- Act as a regression suite so that future changes to the api will be detected.
- Provide a list of supported plugin features by testing features and asserting that they are known to be working.
- Provide documentation of features by recreating user interaction.

#### Feature Tests Creation

Supporting features is detailed workflow is detail in the *Test Framework Section*.

#### Documentation

Documentation forms the final core principle of development. Without documentation that enable users to understand how to use the plugin, there is not much point in development.

### 2.1.2 Workflow

The aim of this section is to describe the desired workflow for a developer

### Porting Strategy

We are following the following strategy for porting the plugin:

1. Write regression test for desired feature.
2. Run the test.
3. Fix the first exception that occurs, and commit the fix.
4. Go back to step 2 until no more exceptions are raised.
5. Do the next 2.6.x release.
6. Listen to feedback from users, and go back to step 1.

The 2.6.x series is designated as purely experimental.

Once enough features have and pass their regression test—i.e. as soon as the new plugin can be considered en par with the old scripts—the code will be refactored and cleaned up, possibly moving some bits out to seperate addons (hull script, morph copy, etc.). The refactor is reserved for the 3.x.x series.

### Generate Documentation

Run the following in a buildenv (Windows) or terminal (Fedora):

```
make html
```

from within the `blender_nif_plugin/docs` folder. The generated API documentation will correspond to the currently installed plugin (*not* your checked out version!) so usually you would install it first.

To view the docs, open `docs/_build/html/index.html`.

## 2.1.3 Development Issues

This document lists a few non-trivial issues that we have come across while porting the plugin from the Blender 2.4x API to the 2.6x+ API.

### Matrices

OpenGL - RHS DirectX - LHS 1 0 0 1 0 0 0 cos(x) -sin(x) -> 0 cos(x) sin(x) 0 sin(x) cos(x) 0 -sin(x) cos(x)

cos(y) 0 sin(y) cos(y) 0 -sin(y) 0 1 0 -> 0 1 0 -sin(y) 0 cos(y) sin(y) 0 cos(y)

cos(z) -sin(z) 0 cos(z) sin(y) 0 sin(z) cos(z) 0 -> -sin(z) cos(z) 0 0 0 1 0 0 1

Memory: [ 0][ 1][ 2][ 3][ 4][ 5][ 6][ 7][ 8][ 9][ 10][ 11][ 12][ 13][ 14][ 15]

DirectX [m11][m21][m31][m41][m12][m22][m32][m42][m13][m23][m33][m43][m14][m24][m34][m44]

OpenGL [0,0][1,0][2,0][3,0][0,1][1,1][2,1][3,1][0,2][1,2][2,2][2,3][0,3][1,3][2,3][3,3]

Access Repr Access Repr [m11][m12][m13][m14] [ 1][ 0][ 0][ ] [00][01][02][03] [ 1][ 0][ 0][ x] [m21][m22][m23][m24] [ 0][ cos][ sin][ ] [10][11][12][13] [ 0][cos][-sin][ y] [m31][m32][m33][m34] [ 0][-sin][ cos][ ] [20][21][22][23] [ 0][sin][ cos][ z] [m41][m42][m43][m44] [ x][ y][ z][ ] [30][31][32][33] [ ][ ][ ][ w]

Matrix multiplication For a given matrices A B C where A * B * C -> D the equivilant D' in the alternate handed system is C' * B' * A'

### Objects

- Vertex groups are accessible via `bpy.types.Object.vertex_groups`, instead of via `bpy.types.Mesh`.

### Meshes : Index Mapping

- Beware that, unlike in blender 2.4x, `bpy.types.MeshFace.vertices` does not return a list of the type `bpy.types.MeshVertex`. Rather `int`s are returned which are index values mapping `bpy.types.MeshFace.vertices` to `bpy.types.Mesh.vertices`, so you need for instance:

```
(b_mesh.vertices[b_vertex_index].co for b_vertex_index in b_face.vertices)
```

  when requiring the actual vertex coordinates of a `bpy.types.MeshFace`.

  This index mapping is also used by attributes such a vertex color, vertex weight, vertex uv

### Meshes

- Beware of the **eeekadoodle dance**: if face indices end with a zero index, then you have to move that zero index to the front. For example (assuming every face is a triangle):

```
faces = [face if face[2] else (face[2], face[0], face[1])
         for face in faces]
```

  before feeding faces to blender.

- It appears that we have to use `bpy.types.bpy_prop_collection.add()` (undocumented) and `bpy.types.bpy_prop_collection.foreach_set()` on `bpy.types.Mesh.vertices` and `bpy.types.Mesh.faces` to import vertices and faces:

```
from bpy_extras.io_utils import unpack_list, unpack_face_list
b_mesh.vertices.add(len(verts))
b_mesh.faces.add(len(faces))
b_mesh.vertices.foreach_set("co", unpack_list(verts))
b_mesh.faces.foreach_set("vertices_raw", unpack_face_list(faces))
```

  After this has been done, uv and vertex color layers can be added and imported:

```
b_mesh.uv_textures.new()
for face, b_tface in zip(faces, b_mesh.uv_textures[0].data):
    b_tface.uv1 = uvs[face[0]]
    b_tface.uv2 = uvs[face[1]]
    b_tface.uv3 = uvs[face[2]]
```

  To import say vertices one by one, use:

```
b_mesh.vertices.add(1)
b_mesh.vertices[-1].co = ...
```

  **Note:** This can be improved by batch importing vertices instead of creating verts one by one.

### Animation

- Ipo's are gone. They are replaced by `bpy.types.Object.animation_data` (see `bpy.types.AnimData`).

### Collision

- Beware of the difference between `bpy.types.Object.draw_bounds_type` and `bpy.types.GameObjectSettings.collision_bounds_type` (accessible via `bpy.types.Object.game`):
  - There is no `'CONVEX_HULL'` `bpy.types.Object.draw_bounds_type`.
  - To identify the collision type to export, we rely exclusively on `bpy.types.GameObjectSettings.collision_bounds_type`. This also ensures that collision settings imported from nifs will work with blender's game engine.

### Bone

- Setting up the parent chile relationship is difficult for a number of reasons
- The `bpy.types.Bone.parent` is a read-only value, only writable by through a `bpy.types.EditBone`.
- Assuming that `bpy.types.Bone` 's have been created and added to an `bpy.types.Armature`
- `bpy.types.EditBone` 's are access via the collection attribute `bpy.types.Armature.edit_bones`, which only exists while in Edit mode.
- EditBones are accessed through `int` indexed rather `str` index b_armatureData.edit_bones[b_child_bone.name].parent = b_armatureData.edit_bones[b_bone.name]

### Strings and Bytes

Generally, we use `str` everywhere, and convert `bytes` to `str` whenever interfacing directly with the nif data.

**Todo:** Add an encoding import/export option.

### Error Reporting

With the older blender 2.4x series, scripts could report fatal errors simply by raising an exception. The current blender series has the problem that *exceptions are not passed down to the caller of the operator*. Apparently, this is because of the way the user interface is implemented. From a user perspective, this makes no difference, however, for testing code, this means that **any exceptions raised cannot be caught by the testing framework**.

The way blender solves this problem goes via the `bpy.types.Operator.report()` method. So, in your `bpy.types.Operator.execute()` methods, write:

```
if something == is_wrong:
    operator.report({'ERROR'}, 'Something is wrong.')
    return {'FINISHED'}
```

instead of:

```
if something == is_wrong:
    raise RuntimeError('Something is wrong')
```

When the operator finishes, blender will check for any error reports, and if it finds any, it will raise an exception, which will be passed back to the caller. This means that we can no longer raise *specific* exceptions, but in practice this is not really a problem.

Following this convention makes the operator more user friendly for other scripts, such as testing frameworks, who might want to catch the exception and/or inspect error reports.

The `io_scene_nif.import_export_nif.NifImportExport` class has a dedicated `error()` method for precisely this purpose.

The list of reports of the last operator execution can be inspected using `bpy.ops.ui.reports_to_textblock()`.

#### Blender API Mysteries

- What is the difference between `'CAPSULE'` and `'CYLINDER'` `bpy.types.Object.draw_bounds_types` (and similar for `bpy.types.GameObjectSettings.collision_bounds_type`)? We are using `'CYLINDER'` at the moment because `'CAPSULE'` is lacking visualisation.
- How do you get the set of all vertices in a `bpy.types.VertexGroup`?

#### Solved

- What is the difference between `bpy.types.MeshFace.vertices` and `bpy.types.MeshFace.vertices_raw`?
- vertices is a collection, accessible in the form vertices.co[0] -> 7
- vertices_raw returns a list of values -> (7,2,0)

### 2.1.4 Naming Conventions

- Stick to the official Python style guide (PEP 8).
- Instances of blender classes start with `b_` whilst instances of nif classes start with `n_`. Examples:
  - `b_mesh` for a blender `bpy.types.Mesh`
  - `b_face` for a blender `bpy.types.MeshFace`
  - `b_vertex` for a blender `bpy.types.MeshVertex`
  - `b_vector` for a blender `mathutils.Vector`
  - `b_obj` for a blender `bpy.types.Object`
  - `b_mat` for a blender `bpy.types.Material`
  - `b_bone` for a blender `bpy.types.Bone`
  - `n_obj` for a generic `pyffi.formats.nif.NifFormat.NiObject`
  - `n_geom` for a `pyffi.formats.nif.NifFormat.NiGeometry`

**Todo:** These conventions are not yet consistently applied in the code.

## 2.2 Plugin Development Setup

The following documentation is targeted towards anyone who is interested in building the latest version of Blender Nif Plugin. This will guide those new through the process of setting up the requirements to build the plugin.

For those interested in developing the Plugin; fixing bugs, adding new features, tests or improving documentation.

Contents:

### 2.2.1 Setting Up the Build Environment

**Warning:** The following instructions are currently a work in progress.

#### Create a Workspace

First, create a directory to be used as your coding directory.

- `C:\Users\<username>\workspace` (Vista/Win 7).
- `C:\Documents and Settings\<username>\workspace` (XP).
- `/home/<username>/workspace` (Linux).

#### Source Code

Once you have created the directories, time to grab the source code. See *source code* to setup source control and download the code repositories. You will also need to download buildenv, it used to manage dependancies, needs to be updated for nix platforms.

#### Install Python 3.4

**Windows**

1. Download Python 3.4.
2. Pick the installer appropriate for your platform, and follow the instructions.
3. Use the default install location. (recommended)

**Fedora**:

```
sudo yum install python3
```

**Ubuntu**:

```
sudo apt-get install python3-minimal
```

This is installed by default on 14.04 or later

#### Environment Variables

**Windows**:

```
set /p BLENDER_ADDONS_DIR=<path_to_blender_addons>
```

**Ubuntu**:

```
set -X BLENDER_ADDONS_DIR=<path_to_blender_addons>
```

#### Install Blender

See *user docs*. Alternatively you can build blender from source *Building Blender from Source*

#### Install Sphinx and Nose

Dependency scripts are available in the install directory.

**Windows** run in buildenv:

```
install_deps.bat
```

**Unix**:

```
install_deps.sh
```

Using software management:

**Ubuntu** run in a terminal:

```
sudo apt-get install python3-nose python3-sphinx
```

**Fedora** run in a terminal:

```
sudo yum install python3-nose python3-sphinx
```

#### Check Installation

To verify everything is installed correctly, start blender, open the internal Python console, and type:

```
import sphinx
import nose
```

You should not get any import errors.

### 2.2.2 Downloading the Source Code

This section will guide new developers through the process of downloading the source code. We use git as to manage and version our source control. It is hosted on github, a popular git hosting platform.

### Install Git

The code is maintained with git. If you are not yet familiar with git, read progit.

**Windows**

We use git bash. Download git bash installer and follow the instructions.

**Fedora**:

```
sudo yum install git
```

**Ubuntu**:

```
sudo apt-get install git
```

### Auto CLRF

- We need to ensure consistency of end-of-file(EOF) markers between unix & windows platforms.
- Locally it will keep your platform specifics EOF, but when you go to push it will update files as necessary
- This avoids unnecessary commits, when your environment is different to the remote.
- Without this option, Git will see the different markers and think that the whole file has been edited.
- Read EOF.
- For Windows-style line endings, use:

  ```
  git config --global core.autocrlf true
  ```

  For Unix-style line endings, use:

  ```
  git config --global core.autocrlf input
  ```

  Either option ensures that all commits in the git history will be stored using Unix-style endings, and that all checkouts (i.e. actual files) will have consistent line endings according to your operating system.

### Setup SSH Keys

Follow the instructions at Github's SSH Page.

---

**Note:**

- If you run into errors while starting the ssh-agent or adding the keys to the ssh-agent try running "eval *ssh-agent -s*".

---

### Create a Github Fork

If you intend to work on the Blender nif plugin, first, you should clone the code on github.

1. If you do not have one yet, create a github account.
2. Set up your git environment.
3. Log in on github.

4. Visit the Blender Nif plugin mothership repository.
5. Click **Fork** (top right corner).

Be sure to read the remaining github help pages, particularly the beginner's guides.

#### Get the Source Code

To get the code, run in a terminal (linux) or in git bash (windows):

```
cd ~/workspace
git clone --recursive git@github.com:<username>/blender_nif_plugin.git
cd blender_nif_plugin
```

We use submodules to maintain external dependencies. This allows us to update to version of the dependency independently of the corresponding project's release cycle.

Fetching the submodules:

```
$ git submodule update --init
```

If you get the following error:

```
fatal: Needed a single revision
Unable to find current revision in submodule path 'pyffi'
```

Run:

```
$ rm -rf pyffi
$ git submodule update --init
```

Optional remote tracking of other developers:

```
git remote add <developer> git://github.com/<developer>/blender_nif_plugin.git
```

### 2.2.3 Building Blender from Source

**As a developer, there are a number of advantages that come from building from source.**

- Avoid having to wait for bug fixes from full releases
- Building the latest version to test compatability early
- Testing new features and how we can integrate with them
- Blender can be built as a python module, which can improve IDE integration.

The Blender code repo is also managed by git, allowing ease of integration into our workflow. There are some additional prerequisite utilities that need to be installed first.

See - https://wiki.blender.org/wiki/Building_Blender

The repo comes with scripts which will package up the plugin for use

### 2.2.4 Building the Blender Nif Plugin

The Blender Nif Plugin is a python project which can be manually put into the Blender add-ons directory. The repo provides a set of scripts which allows creating of a zip file which can be loaded into Blender Addon Manager.

**Note:** Ensure that you have installed the prerequisite dependencies using install_deps scripts

#### Build Zip

To build the plugin from a git checkout, run the following

```
cd ./blender_nif_plugin/install
makezip.bat
```

#### Install

To install the plugin from a git checkout, run the following

```
cd ./blender_nif_plugin/install
install.bat
```

or from a terminal (Linux | Ubuntu):

```
cd ./blender_nif_plugin/install
sh ./install.sh
```

### 2.2.5 Interactive Development Environment

PyCharm IDE is the preferred way to allows us maintain a unified workflow for general file manipulation, repo management, python scripting, and hooks into Blender's debugging server.

#### Install PyCharm

**Windows**

1. Download PyCharm

**Fedora**, simply run:

```
sudo yum install eclipse eclipse-egit eclipse-pydev
```

**Ubuntu**, simply run:

```
sudo snap install [pycharm-professional|pycharm-community] --classic
```

When starting pycharm, Open the workspace folder. If you followed the instructions, you should have cloned the code into `/home/<username>/workspace` PyCharm will automatically recognise this as a Git repo.

#### Debugging

The Blender Nif plugin code comes with built-in pydev hook to allow connection by a Remote Python Debug Server. This allows run-time debugging; watching the script execute, evaluate variables, function call stack etc.

#### Debugger

..TODO

Happy coding & debugging.

## 2.3 API Documentation

The following documentation contains high-level overview of components and submodules. The sub-module documentation is generated automatically using Sphinx.

Contents:

### 2.3.1 User Interface

- The user first activates the addon via **File > User Preferences > Addons**.
- This triggers the `register()` function, which adds the `NifImportUI` and `NifExportUI` operators to the **File > Import** and **File > Export** menus.
- These operators are integrated within the user interface, and their responsibility is to allow the user to configure the import and export properties.
- They delegate the actual import and export to the `NifImport` and `NifExport` classes.

### 2.3.2 Indices and tables

- genindex
- modindex
- search

## 2.4 Test Framework

The test framework is a series of tools to automate the verification process of the Blender Nif Plugin It aims to unify the various levels of testing that should be performed to release the plugin

- Unit
- Functional
- Integration
- Regression
- Performance

For an overview of each level see the :ref: *design section <development-testframework-design>*

### 2.4.1 Install

To install the plugin from a git checkout, run the following

```
cd blender_nif_plugin/install
install.bat
```

or from a terminal (Linux | Ubuntu):

```
./blender_nif_plugin/install
sh ./install.sh
```

### 2.4.2 Running Tests

To run all tests, run the following in a buildenv (Windows):

```
blender-nosetests.bat
```

or from a terminal (Linux | Ubuntu):

```
sh ./blender-nosetests.sh
```

from within the `blender_nif_plugin/testframework/` folder.

Each test resides as a python file in the `blender_nif_plugin/testframework/test/` folder. To run a particular test only, specify the file as an argument; for instance:

```
blender-nosetests.bat test/geometry/trishape/test_geometry.py
```

Actually, all command line arguments of `nosetests` apply. For example, to abort on first failure:

```
blender-nosetests.bat -x
```

For more details, run:

```
blender-nosetests.bat --help
```

- The tests will run on the currently installed plugin (*not* your checked out version!) so usually ensure you reinstall after making edits to add-on files.
- Beware that the output can be rather verbose, so you may have to scroll quite a bit to see the relevant backtrace.

Also see the nose manual.

#### TestFramework Design Documentation

This section provides a high level overview of the design of the Blender Nif Plugin. It is geared towards anyone interested in delving into developing the Blender Nif Plug-in or knowing more about the development process.

Contents:

### Integration Test Design

#### Test Template

For every feature, first, a test should be written.

The following process is followed:

1. Create a new python file to contain the feature test code. For example, if the feature concerns *x_feature*, the test case would be stored in `testframework/test/../../test_x_feature.py`.
2. Derive the test class from `test.SingleNif` and name it `TestXFeature`. `test.SingleNif` is a designed upon the template pattern and takes care of all execution, loading of files and clean up.

   **Note:** A template for a test class is available in `testframework/test/template.py`.

3. Each test needs to overwrite the following methods from `test.SingleNif`
   - The `n_create_data()` used to create the physical .nif
   - The `n_check_data()` used to test the data of physical .nif
   - The `b_create_data()` used to mimic how the user would create the objects.
   - The `b_check_data()` check blender data, used to check before export and after import.

These methods are intended to be delegate function, calling external methods in either n_gen_xxxx or b_gen_xxxx files.

- We move all the code to external modules for code reuse code to avoid importing other tests, avoiding tests to be re-run unnecessarily.

#### Test Implementation

Each test has two paths of execution one importing/export a python generated nif, the other exporting/importing a nif created by mimmicking how a user would create it. Each test can be comprised of multiple required features and a lego block-building approach should be taken to building the nifs.

1. Recreate the nif using python:
   - Create a nif (say in nifskope, or with the old blender nifscripts). Take care to make the file as simple as possible.
   - Use pyffi's `dump_python` spell to convert it to python code.
   - The `n_create_data()` method of the class will call the methods from `n_gen_x_feature` module to construct the physical nif. It may be required to call other reusable functions from other tests. Reusable functions should be created from the dump_python and stored in `testframework/test/../../n_gen_x_feature.py`.
   - Write code to test the desired features of the physical. The `n_check_data()` method will call the methods from `n_gen_x_feature` module to check the nif data.

1. Recreate the feature within blender, using user functions:

- Write Python code which recreates the corresponding data in the blender scene in `testframework/test/../../b_gen_x_feature`.
- Where possible make the test case as simple as possible. For instance, use primitives readily available in blender. This code goes in the `b_create_data()` method of the test class.

- Document the feature in `docs/features/x_feature.rst` as you write `b_create_data()`: explain what the user has to do in blender in order to export the desired data, and where in blender the data ends up on import.
- Write Python code which test the blender scene against the desired feature: `b_check_data()` method of the test class.

1. Now implement the feature in the import and export plugin, until the regression test passes.

That's it!

### Execution Order

The tests will run like this:

### User Export

1. `b_create_data()` to create the scene, saved to `test/autoblend/../../x_feature_userver.blend`
2. `b_check_data()` to check it before export
3. Export the nif to `` `test/nif/../../x_feature_export_pycode.nif ``
4. `n_check_data()` to check exported nif.

### User Import

1. import the exported nif, saved to `test/autoblend/../../x_feature_userver_reimport.blend`
2. `b_check_data()` tests the imported scene.

If the above tests run, then we are in pretty good shape as we can verify import and export work in isolation

### Python generated Import / Export

1. Starts by `n_create_data()` creating a physical nif `test/nif/../../x_feature_py_code.nif`.
2. `n_check_data()` is called to ensure nif is correct before importing.
3. Nif is imported into blender, the scene is saved to `test/autoblend/../../x_feature_pycode_import.blend`
4. `b_check_data()` is called on imported scene to verify scene data.
5. Nif is exported to `test/nif/../../x_feature_export_pycode.nif`
6. `n_check_data()` on exported nif to verify nif data.

This ensures data integrity both at Blender level and at nif level.

### TestFramework API Documentation

The following documentation contains high-level overview of the test framework. The sub-module documentation is generated automatically using Sphinx.

Contents:

### Indices and tables

- genindex
- modindex
- search

### Continuous Integration Build Server

The following section is both for internal reference for the build server used by the Niftools team. We encourage developers to build their own; it provides a fast feedback loop, encourages you to run test and fix them when they get broken. We use jenkins as our build server

Installed Plug-ins

- EnvInject Plugin : Used to inject env variables generated by Buildenv
- Git Client & Git Plugins : Used to grab repo
- Hudson Build-publisher Plugin : Pre-installed
- Jenkins Cobertura Plugin : Used to publish coverage reports

**Optional Plug-ins**

- IRC Plugin : IRC bot which allows you trigger builds remotely

Project Name

```
blender_nif_plugin
```

Git Repository

```
git://github.com/niftools/blender_nif_plugin.git
```

Branches

```
develop
```

Build Triggers

```
Poll SCM
Build Periodically
```

Inject Environment variables

```
..<path_to_jenkins>/.jenkins/workspace/bin/blender.properties
```

### Build Steps

Build

```
> cd install
> DEL *.zip
> makezip.bat
```

Unit Test

```
> cd testframework
> blender-nosetests.bat
--with-xunit
--xunit-file=reports\unit.xml
--cover-xml
--cover-package=io_scene_nif unit
```

Integration Test

```
> cd testframework
> testframework\blender-nosetests.bat
--with-xunit
--xunit-file=testframework\reports\integration_test.xml
--cover-package=io_scene_nif
--cover-xml-file=testframework\reports\integration_test_coverage.xml
testframework\integration
```

Nightly

```
> cd install
> ren *.zip blender_nif_plugin_%BUILD_NUMBER%_%BUILD_ID%.zip
> xcopy /K /F  "*.zip" "%build_folder%\blender_nif_plugin\nightly\"
```

### Post Build Actions

Publish Cobertura Coverage reports

```
testframework/reports/*.xml
```

Publish XUnit test reports

```
testframework/reports/*.xml
```

## 2.4.3 Indices and tables

- genindex
- modindex
- search

## 2.5 Documentation

Documentation of the Blender Nif Plugin is of equal importance to having a clean, functional code base and automated tests. The plugin is intended to be used by content creators. The main causes for inability to use it correctly are. - Functional bugs - Design and usabilty issues - Poorly documented features.

The first two issues can be fixed on a priority basis, unless things are documented to reflect there are known issues or the workflow is outlined correctly users will be able to use the plugin correct. This results in poor user experience and more work for developers as they will have to respond to support requests.

### 2.5.1 Convention

This chapter outlines the convention for documentation based on the Python style guide

Section headers are created by underlining (and optionally overlining) the section title with a punctuation character, at least as long as the text:

==========
This is a heading
==========
Normally, there are no heading levels assigned to certain characters as the structure is determined from the succession of headings. However, for the Python documentation, here is a suggested convention:

# with overline, for parts
* with overline, for chapters
=, for sections
-, for subsections
^, for subsubsections
“, for paragraphs

## 2.6 Indices and tables

- genindex
- modindex
- search

CHAPTER 3

# Authors

## 3.1 Current developers

- neomonkeus (neomonkeus@users.sourceforge.net)

## 3.2 Former developers

- Amorilia (amorilia@users.sourceforge.net)
- Brandano (brandano@users.sourceforge.net)
- Ghostwalker71
- m4444x

## 3.3 Contributors

- Aaron1178
- Alphax (alphasigmax@users.sourceforge.net)
- Arcimaestro
- Arthmoor
- Artorp
- Deedes
- Eli2
- Entim
- Eugenius-v

- Fritz_fretz
- GandaG
- Hanaisse
- Kikaimegami
- Lhammonds
- malo
- mgm101
- Pacificmorrowind (pacmorrowind@users.sourceforge.net)
- Pentinen
- shon
- Tamira
- Thedaywalker
- Tijer
- zin

CHAPTER 4

# Contributing

## 4.1 Contributions Overview

The project is always looking for people to contribute to the project. The following rules are guidelines for the various forms of contributions.

### 4.1.1 Code Contribution Process

Contributions should always be made through GitHub's Pull Request (PR) feature, branched from latest *develop* branch. This repo comes with a template for Pull Requests and should be fully completed.

Once the PR is submitted, the @niftools/blender-nif-plugin-review team is automatically notified and will start a discussion process. This can include ensuring the commit is beneficial to the project as a whole, is appropriately scoped, discuss any issues with the design, ensure all requirements are met. This is an iterative process; additional commits can be appended at any time based on PR discussion to the source branch and will be automatically included in the PR.

**There are three conditions any contribution to the codebase should respect:**

- Clean, commented implementation that doesn't break existing functionality
- Tests to prove the implementation works and regression is still green
- Appropriate documentation updates

These are not exclusive conditions for a contribution to be accepted, but generally enforced as close as possible to ensure quality. Merging of the PR is at the discretion of the @niftools/blender-nif-plugin-merge team.

#### Code

**General rule of thumb is to see what has gone before, but if your way is better, ensure you demonstrate that to the review team.**

- Adhere to conventions, see out main documentation for developers.

- Code should be as readable as possible.
- Comments should help break down complex behavior. If the behavior is too complex, group the code into logical chunks, or extract to functions for easier to understand code.
- Docstrings to describe the intention, not implementation. Self describing code is good and may provide functionality, but useless if it doesn't do what is actually needed.

#### Testing

**Our template includes sections relating to ensure appropriate testing has been completed.**

- If you change the behavior of part of the code, you should try to change the relevant tests to adapt to the new behavior.
- If a test fails, it has to be fixed either in the test or the code.

When possible, for every API additions add either unit or integration tests. The template requires documenting any tests changes in the Pull Request. Expect questions about any test updates or deletions.

#### Documentation

**Documentation is used by both users and developers and is published.**

- Any updates that add, remove or modify behaviour require corresponding updates to ensure that user can use the functionality
- Supporting developer documentation should be updated with relavent changes, design decisions, etc.

### 4.1.2 Reviewer Contribution

**The review group is to ensure that contributions are openly discussed, people can participate in the development process and en**

- If you are interested in joining the review group to give your input on contributions, talk to the @niftools/core team to be added as reviewer.
- You will automatically recieve notificiations for any PR related to this repo.

### 4.1.3 Release process

**When all issues in a milestone are complete, a release branch is created from the *dev* branch, tagged, log updates and PR done t**

- The project uses the Python versioning scheme - https://www.python.org/dev/peps/pep-0440/#public-version-identifiers
- A release will always include an accompanying version change, but the reverse is not true : the version can change without a release.

After being merged, a release will be officially released using the GitHub release system.

CHAPTER 5

# Changes

## 5.1 Version 2.6.0a3 (3 Jan 2015)

- Migrated to Bmesh API
- Fix uv layer detection
- Additional material properties support (alpha, specular, stencil, wire).
- Add support for NiTexturingProperty (diffuse, bump, normalmap, specular and glow).
- Fix crash when combine shapes is enabled (reported and fix contributed by Aaron1178)
- Fix issue with material texture blend type importing (reported and fix contributed by mgm101).
- Added experimental vertex color support.
- Collision support:
  - Basic BhkShapes Cube, Sphere, Cylinder.
  - Convex Vertex, NiPacked, NiTriStrips Shapes.
  - Bound Box support.
- Subsystem separation (collision, armature, material, texture).
- Bundle pyffi with scripts.
- Distribute zip that can be used with Blender's built-in installer.
- Extensive work on the testing framework:
  - Tests created based on new features.
  - Re-organised tests into per feature, generated test nifs.
  - Inheritance based checks now functioning.
- **Documentation vastly improved.**
    - Feature documentation

  - Developer documentation, api auto-doc and workflow

## 5.2 Version 2.6.0a0 (20 Nov 2011)

- Initial port to Blender 2.60a:
  - geometry (NiTriShape)
  - materials (NiMaterialProperty)
  - uv textures (NiTexturingProperty)
- Upgraded to sphinx to generate documentation.
- Upgraded to nose for testing.

## 5.3 Version 2.5.8 (30 Oct 2011)

- Fix for collision objects that have no vertices (see issue #3248754, reported by rlibiez).
- Fix for export of convex collision shapes bundled together in a list (see issue #3308638, reported by Koniption).
- Updated installer to get the Blender 2.49b installer if Blender is not yet installed (this avoids confusion with Blender 2.5x+ being the default on the official download page).

## 5.4 Version 2.5.7 (26 March 2011)

- added rubber material (reported by Ghostwalker71)
- havok material name bugfix
- fixed issue with disfunctional havok constraints in ANIM_STATIC, CLUTTER, and BIPED layers (reported and fix contributed by Koniption)
- also import BSBound bounding box on dummy scene node
- fixed BSBound import and export scale (see issue #3208935, reported and fix contributed by neomonkeus)

## 5.5 Version 2.5.6 (4 February 2011)

- fix import in case skin instance has empty bone references (fixes pyffi issue #3114079, reported by drakonnen)
- updated for PyFFI 2.1.8
- fix export of bezier curve animation (reported by arcimaestro)
- split multimaterial mopps into different NiNodes so we only have single material mopps; this works around the usual mopp issues (reported by neomonkeus)
- new export option for Bully SE; sane settings are automatically selected
- improved support for Divinity 2 nifs (reported by pertinen)

## 5.6 Version 2.5.5 (18 July 2010)

- fixed bone priority import for L and R bones (reported by Da Mage)
- updated for PyFFI 2.1.5
- fixed NiCollisionData import (reported by LordOfDragons)

## 5.7 Version 2.5.4 (28 Mar 2010)

- fixed bone priority export for L and R bones (reported by Kilza)
- fixed morph base key name import (reported by LHammonds)
- fixed morph base key to have no float data (reported by LHammonds)
- improved export of controller start and stop times (reported by LHammonds)
- fixed consistency type on NiGeometryData to be CT_VOLATILE when exporting morphs; this fixes for instance bow exports (reported and fix suggested by LHammonds, based on Nicoroshi and Windy's bow tutorials)

## 5.8 Version 2.5.3 (19 Mar 2010)

- import and export NiLODNodes as empty with LODs as children and properties to set extents
- added material color controller import and export (request and test files by Alphax)
- added vis controller import and export (request and test files by Alphax)
- fixed some controller imports in case controller block had no data
- improved Fallout 3 skeleton.nif import
- fixed bhkCapsuleShape export with identical points by converting it to a bhkSphereShape (reported by ghostwalker71)
- warn if mopp is exported for non-static objects, as these may not function properly in-game (reported by mc.crab)
- added option to use NiBSAnimationNode when exporting animated branches for Morrowind (suggested and tested by TheDaywalker)

## 5.9 Version 2.5.2 (20 Feb 2010)

- configurable game paths for test suite
- fixed display of alpha channel in textured faces (reported by vurt2, fixed by Alphax)
- weight squash script can now limit the number of bone influences per vertex (requested by Growlf)
- disabling combine shapes import option results in xbase_anim type nifs to import clothing slots as bones (fixes transform issue reported by Arcimaestro Anteres)
- added regression test and workaround for duplicate shape keys during import: only the first is read, and duplicates are ignored (e.g. Fallout 3 skeleton.nif HeadAnims:0)

- added regression test and workaround for corrupt translation keys in Fallout 3 interpolators (e.g. Fallout 3 h2haim.kf, reported by Malo)
- added experimental .kf export for Freedom Force and Freedom Force vs. the 3rd Reich
- fixed interpolator bug with bhkBlendControllers when exporting kf files for creatures with bones that have havok blocks attached (reported by Spiderpig)
- added alpha controller import; export was already implemented (requested and test files provided by Alphax)
- fixes/improvements to animation import and export
  - full support for import/export of animation priority
  - autoset target name to bip02 if the armature has such a bone
  - new option to manually set the target name on export
  - new option to bulk set the animation priority
  - skip NiBSplineInterpolators on import; not fully supported and if not skipping was causing a fatal error
- fix for bhkNiTriStripsShape import
- added experimental import and export of Empire Earth II meshes
- fixed bhkCapsuleShape import with identical points (reported by ghostwalker71)

## 5.10 Version 2.5.1 (10 Jan 2010)

- updated for pyffi 2.1.0
- fixed stencil property export for Fallout 3
- Morrowind bounding box import and export
- import and export, via object properties per object, of havok object
  - material
  - collision layer
  - motion quality
  - motion system
  - mass
  - col filter
- import and export, via object properties per object, of havok constraint
  - min angle
  - max angle
  - friction
- object rotation animation import bugfix (reported by Arcimaestro Anteres, fixes for instance Morrowind animated creature imports)
- fix for Fallout 3 NiGeomMorpherController (shape key) export (reported by Bleolakri)
- pep8 fixes
- installer detects Python 64 bit, and complains about it

- increased resolution of vertex coordinates to 1/1000 (from 1/200) on import and export (fixes issue #2925044 reported by EuGENIUS).
- added support for Atlantica and Howling Sword import and export

## 5.11 Version 2.5.0 (22 Nov 2009)

- attempt to fix invalid matrices in bone extra text buffer rather than raising a mysterious exception (reported by PacificMorrowind)
- import and export Oblivion morph controller animation data (reported by LHammonds, addition testing and bug reports by PacificMorrowind)
- import extra nodes as empties
- extra nodes are now imported by default (suggested by PacificMorrowind)
- various object animation import and export fixes (reported by LHammonds and Tijer)
- enable flattening skin in the export gui when 'geometry only' is selected, for Oblivion and Fallout 3 (contributed by PacificMorrowind)
- civ4 and Sid Meier's Railroads NiNode and NiTriShape flags are now set to 16 (reported by Tijer)
- on import, set alpha to 0.0 if NiAlphaProperty is present (so it gets re-exported) even if no textures with alpha channel are found; this fixes an issue with Sid Meier's Railroads (reported by Tijer)
- export NiAlphaProperty threshold 150 for Sid Meier's Railroads (reported by Tijer)
- export RRT_NormalMap_Spec_Env_CubeLight shader for Sid Meier's Railroads (reported by Tijer)
- force TSpace flag to be 16 for Sid Meier's Railroads and Fallout 3 (reported by Tijer and Miaximus)
- fixed windows installer & installer scripts to install to the dirs currently expected by blender (contributed by PacificMorrowind)
- import and export egm morphs (with aid of Scanti and Carver13)
- added new experimental "morph copy" script (under scripts->mesh)
- stitch strips for Fallout 3 by default (reported by Miaximus)
- fixed texture path bug (reported by elitewolverine)

## 5.12 Version 2.4.12 (23 Oct 2009)

- warn and ignore object animation on skinned meshes, instead of raising a mysterious exception (reported by vfb)
- added Zoo Tycoon 2 .kf export
- added dialogue requesting animation sequence name for .kf export (contributed by PacificMorrowind)
- added preset for Oblivion OL_ANIM_STATIC objects (see issue #2118370 reported by apwsoft; fix discovered by PacificMorrowind)
- export XYZ rotations for object animations instead of converting to quaternions (reported by Artorp)
- set bhkCollosionObject flag to 41 instead of the default 1 for animated (OL_ANIM_STATIC) objects (reported by Artorp)
- updated readme with detailed install instructions

## 5.13 Version 2.4.11 (28 Sep 2009)

- added NeoSteam import and export support
- warn on corrupt rotation matrix, rather than raising an exception
- bug fix in case (corrupt) root block has no name attribute
- fix for collision export with very small mass (contributed by PacificMorrowind, see issue #2860536)

## 5.14 Version 2.4.10 (22 Jul 2009)

- windows installer updated for Python 2.6 and PyFFI 2.0.1.
- set affected node list pointer on Morrowind environment map (contributed by Alphax)
- use Blender's texture dir on import (contributed by puf_the_majic_dragon)

## 5.15 Version 2.4.9 (20 Jun 2009)

- test and fix for NiKeyframeController target in Morrowind xkf files (reported by arcimaestro, see issue #2792951)
- test and fix for NiKeyframeController flags import and export: the nif cycle mode is mapped onto the blender ipo curve extrapolation mode (reported by arcimaestro, see issue #2792951)
- test and fix for anim buffer out of range exception - the exporter will now only warn about it but continue with export anyway (reported by arcimaestro, see issue #2792952)
- fixed bug when importing extra bones which were parented on a grouping bone (for instance Morrowind atronach_frost.nif, where Bone01 is parented to Weapon, which groups the geometry Tri Weapon)

## 5.16 Version 2.4.8 (3 Jun 2009)

- fixed bug in hull script (reported by Drag0ntamer, fixed by Alphax)

## 5.17 Version 2.4.7 (4 May 2009)

- fixed bug where "apply skin deform" would apply it more than once on geometries that are linked to more than once in the nif
- new option to import extra nodes which are not bone influences as bones (reported by mac1415)
- bugfix for Euler type animation import
- max bones per partition now defaults to 18 for civ4 (reported by mac1415)
- updated for pyffi 2.0.0
- moved advanced import settings to new column (reported by Alphax)
- inverted X and Y offset UV Ipo channels on import and export (reported by Alphax)
- added support for civ4 shader textures (reported by The_Coyote)

- new option to control export of extra shader textures for civ4 and sid meier's railroads (reported by The_Coyote)
- if extra shader textures are exported, then tangent space is generated (reported by The_Coyote)
- fixed scaling bug if scale was not 1.0 in certain cases (such as civ4 leaderheads, reported by The_Coyote)
- realign bone tail only is now the import default (slightly better visual representation of bones in complex armatures such as civ4 leaderheads)

## 5.18 Version 2.4.6 (23 Apr 2009)

- import and export of Morrowind NiUVController/NiUVData i.e. moving textures (with help from Axel, TheDaywalker, and Alphax)

## 5.19 Version 2.4.5 (21 Apr 2009)

- another import fix for names that end with null character
- warn on packed textures instead of raising error (reported by augbunny)
- Morrowind:
  - rebirth of the 'nif + xnif + xkf' option for Morrowind (reported by axel)
  - improved import of nifs that have multiple skeleton roots (such as the official skin meshes, and various creatures such as the ice raider)
  - new import option to merge skeleton roots (enable!)
  - new import option to send detached geometries to node position (enable!)
- Fallout 3:
  - now imports and exports the emitMulti value in the shader emit slider (up to a factor 10 to accomodate the range) and stores the emissive color as Blender's diffuse color (reported and tested by mushin)
  - glow texture import and export (reported and tested by mushin)

## 5.20 Version 2.4.4 (2 Apr 2009)

- import option to disable combining of shapes into multimaterial meshes (suggested by Malo, and contributed by Alphax)
- importing a nif with an unsupported root block now only gives error message instead of raising an exception (reported by TheDaywalker)
- fixed fallout 3 import of packed shapes (such as mopps)

## 5.21 Version 2.4.3 (7 Mar 2009)

- further fixes for fallout 3
  - new options in export dialog for shader flags and shader type (thanks to malo and nezroy)
  - new option to disable dismember body part export (sickleyield)

- text keys imported also if they are not defined on the scene root (reported by figurework)

## 5.22 Version 2.4.2 (15 Feb 2009)

- materials whose name starts with "noname" (such as those that are imported without a name) will have no name in the nif; this fixes some issues with Fallout 3 (reported by malo)
- import fix for names that end with null character (reported by alphax)
- if not all faces have a body part, they will be selected on export to make it easier to identify them; error message has been improved too (reported by malo)
- meshes without vertices are skipped; so they no longer give mysterious error messages (reported by malo)

## 5.23 Version 2.4.1 (2 Feb 2009)

- Fallout 3 BSShaderXXX blocks are no longer shared to avoid issues with the engine
- NiSourceTexture improvements:
  - pixel layout exports as "6" (DEFAULT) for versions 10.0.1.0 and higher
  - use mipmaps exports as "1" (YES)
- Sid Meier's Railroads:
  - new regression test
  - fixed import and export of specular color
  - fixed alpha flags export
  - automatic integer extra data export for shader texture indices
  - automatic export of RRT_Engine_Env_map.dds and RRT_Cube_Light_map_128.dds shader texture slots
  - import of extra shader textures, using extra integer data to find the right texture slot
  - bump (i.e. normal), gloss (i.e. spec), and reflection (i.e. emsk) are exported into the extra shader slots instead of in the regular slots
- minor cleanups in the code

## 5.24 Version 2.4.0 (25 Jan 2009)

- switched to using the standard logging module for log messages
- improvements for multi-material mopp import and export (but not entirely functional yet)
- improved self-validating bind position algorithm
  - geometries are transformed first to a common bind pose (if it exists, a warning is issued if no common bind pose is found) - some misaligned geometry pieces will now be aligned correctly with the armature, this is most noticable with Morrowind imports
  - bone nodes are transformed to bind position in two phases, to reduce rounding errors - some bones that were not sent to the bind pose with the older algorithm will now be correct
- better Fallout 3 export options

- added export of Fallout 3 tangent space
- added export of Fallout 3 BSShaderPPLightingProperty for textures
- body parts can now be imported and exported via vertex groups
- fixed RuntimeError when importing mesh without faces

## 5.25 Version 2.3.13 (18 Nov 2008)

- better error message if mesh has bone vertex group but no weights
- improved Civ IV bone flags export (0x6 for intermediate bones, 0x16 for final ones)
- support for double sided meshes via NiStencilProperty and Blender's double sided flag
- NiAlphaProperty flags now defaults to 0x12ED (more useful to modders)
- load bone pose script now works again with saved poses from older blends
- fixed numControlPoints attribute error when importing some kf files such as bowidle.kf (reported by Malo)
- fallout 3 import (very experimental)

## 5.26 Version 2.3.12 (24 Oct 2008)

- activated CivIV kf file export (uses Oblivion style kf, experimental!)
- added option to disable material optimization (prevents "merging")

## 5.27 Version 2.3.11 (19 Oct 2008)

- fix for fresh skeleton import into blends imported with older script versions (again reported by periplaneta)

## 5.28 Version 2.3.10 (18 Oct 2008)

- fix for skin exports from blends imported with older script versions (reported by periplaneta)

## 5.29 Version 2.3.9 (12 Oct 2008)

- improved installer to point to Python 2.5.2 instead of Python 2.6 if Python installation is not found
- improved the test suite
  - allow comparison between imported and exported nif data
  - exported skinning data is now tested against imported skinning data
- added common base class for importer and exporter, for code sharing
- fixed bone correction application which would fail under certain circumstances
- epydoc documentation can now be generated and is included with installation

## 5.30 Version 2.3.8 (27 Sep 2008)

- convert Bip01 L/R xxx to Bip01 xxx.L/R on import, and conversely on export (contributed by melianv, issue #2054493)
- fix for multimaterial geometry morph (shape key) import and export
- show versions of scripts, blender, and pyffi, in import/export dialog (issue #2112995)
- new export dialog options to determine Oblivion weapon location as NiStringExtraData Prn value (issue #1966134)

## 5.31 Version 2.3.7 (25 Aug 2008)

- fixed export of cylinder radius on scaled objects

## 5.32 Version 2.3.6 (19 Aug 2008)

- added import of bhkNiTriStripsShape collisions
- fix for exception when mixing mopps with other primitive shapes
- updated deprecated ipo and curve methods in keyframe export code
- improved FPS estimation on import
- check ipo curve completeness on export (solves the "NoneType has no evaluate attribute" problem)
- fixed scale keys import and export

## 5.33 Version 2.3.5 (25 Jul 2008)

- quick bug fix if you had multiple materials in your mopp

## 5.34 Version 2.3.4 (24 Jul 2008)

- fix for megami tensei imagine collision import
- on merge, do not skip keyframe controller block if the controller is not found in original nif file; instead add a controller to the node in the nif file
- installer fixes for Vista and Blender 2.46
- updated for PyFFI 1.0.0, which includes the new mopp generator based on havok's recently released SDK
- removed mopp option from export config dialog (they are now always generated)
- preserve the "skin", "dynalpha", … material names
- fixed material merge bug
- fix for nif imports with more than 16 materials per mesh (the materials will not be merged in that case)

## 5.35 Version 2.3.3 (May 27, 2008)

- updated installer to make sure PyFFI 0.10.9 is installed

## 5.36 Version 2.3.2 (May 27, 2008)

- B-spline animations are now also imported
- new scripts to save and load current pose of bones to a text buffer (this is useful when changing existing animations and starting/ending pose must be copied over from an existing animation)
- transform controller and interpolator also exported on the Bip01 node on Oblivion skeleton exports
- exporter no longer creates a NiTextKeyExtraData block on skeleton exports

## 5.37 Version 2.3.1 (Apr 13, 2008)

- new script to set bone priorities on multiple bones at once
- Oblivion skeleton import and export including havok and constraints
- also import collision on scene root
- new settings in export dialog to set material and extra havok presets for creature and weapon
- support for NiWireframeProperty via material WIRE mode
- furniture marker export
- prevent merging of EnvMap2 materials with other materials
- import of type 2 and 3 quaternion rotations
- import and export of BSBound bounding boxes for creatures
- many other minor enhancements

## 5.38 Version 2.3.0 (Mar 30, 2008)

- Import/Export: experimental support for Oblivion animation
  - added keyframe file selection to import dialog
  - kf file is merged with nif tree on import
  - includes text keys import from kf file
  - length 1 animations are exported as interpolators without further transform data, and interpolators without further transform data are imported as length 1 animations
  - bone priorities via NULL bone constraint name ("priority:xx")
  - fixed euler rotation animation import (contributed by ahkmos)
  - bspline data is skipped on import
  - only tested on character animations (skeletonbeast.nif + any of the character/_male keyframe animations that don't contain bsplines)

- install.bat for quick windows installation

## 5.39 Version 2.2.11 (Mar 21, 2008)

- Export: NiVertexColorProperty and NiZBufferProperty blocks for Sid Meier's Railroads

## 5.40 Version 2.2.10 (Feb 26, 2008)

- Export: fix for bug in reflection map export

## 5.41 Version 2.2.9 (Feb 22, 2008)

- Import/Export: support for billboard nodes via TRACKTO constraint
- Import: re-enabled embedded texture support (they are saved to DDS)

## 5.42 Version 2.2.8 (Feb 11, 2008)

- Export: more informative error messages if mesh has no uv data and if texture of type image has no image loaded
- Export: fixed NiGeomMorpherController target

## 5.43 Version 2.2.7 (Jan 11, 2008)

- Export: fixed exception when mesh used material with vcol flags enabled but without any vertex colors present
- Import: strip "NonAccum" from name when checking for node grouping
- Import: fixed misaligned collision boxes (sometimes you still have to switch to edit mode and back to align them correctly, seems to be a Blender bug)

## 5.44 Version 2.2.6 (Jan 8, 2008)

- Installer: fixed required PyFFI version

## 5.45 Version 2.2.5 (Dec 18, 2007)

- Export: fixed bug in uv map export with smooth objects

## 5.46 Version 2.2.4 (Dec 10, 2007)

- Import: fixed face orientation of imported bhkPackedNiTriStripsShapes
- Import: also import collisions of non-grouping NiNodes

## 5.47 Version 2.2.3 (Dec 8, 2007)

- Import/Export: added support for gloss textures (use MapTo.SPEC)
- Import/Export: added support for dark textures (use MapTo.COL and blendmode "darken")
- Import/Export: added support for detail textures (add a second base texture, that is, MapTo.COL)
- Import/Export: added support for multiple UV layers
- Import: removed broken pixel data decompression code, so recent nif versions with embedded textures can import (e.g. the copetech nifs)

## 5.48 Version 2.2.2 (Dec 2, 2007)

- Import/Export: support for Morrowind environment maps and bump mapping via NiTextureEffect blocks (set Blender Map Input to "Refl" for the NiTextureEffect texture, see release notes for more details)
- Import/Export: support for the bump map slot (Map To "Nor" in Blender)
- Import: fixed a bug which caused material duplication if materials were shared between more than one NiTriShape/NiTriStrips block
- Import: various small code improvements

## 5.49 Version 2.2.1 (Nov 27, 2007)

- Import: havok blocks (still experimental, but seems to work on most nifs)
- Export: use bhkRigidBody instead of bhkRigidBodyT
- new tester for Blender import and export of havok related blocks
- fixed a bug in the uninstaller (it would not remove the weightsquash script)

## 5.50 Version 2.2.0 (Nov 19, 2007)

- Export: new settings for Oblivion to control rigid body parameters and material
- Export: calculation of mass, center of gravity, and inertia tensor in rigid body, which is useful for non-static clutter
- Config: refactored the config gui to get rid of most geometry parameters when drawing the gui
- updated hull script for quickly creating approximate convex bounding shapes
- the hull script will only hull selected vertices when you run the script in edit mode

## 5.51 Version 2.1.20 (Nov 3, 2007)

- Import/Export: updated for PyFFI 0.6
- Export: ignore lattices when checking for non-uniformly scaled objects
- Export: ignore name when avoiding duplicate material properties
- Test: added babelfish and oblivion full body import/export tests

## 5.52 Version 2.1.19 (Oct 26, 2007)

- Import/Export: emulate apply mode via Blender's texture blending mode

## 5.53 Version 2.1.18 (Oct 25, 2007)

- Export: recycle material, alpha, specular, and texturing properties

## 5.54 Version 2.1.17 (Oct 23, 2007)

- Test: unselect objects when running each test (prevents duplicate exports)
- Import: new option to import bones with original nif matrices (useful in some cases where you do not want to bother with the correction matrices)
- Import: some minor optimizations and code cleanups
- Import: changed some lists to generators to save on memory
- Import: fixed trivial bug in get_blender_object
- Export: improved progress bar
- Export: warn when skin partition settings could be improved on Oblivion export
- Export: check blender objects on non-uniform scaling before export so you do not need to wait too long before the scripts complain about it

## 5.55 Version 2.1.16 (Oct 21, 2007)

- Import: inform about name of Blender object and nif block when losing vertex weights
- Import: update scene even if import fails
- Import: fixed error with parentship if you imported a skeleton without selecting anything
- Import: new experimental option for importing meshes and parenting them to the selected armature (it seems to work pretty well for Oblivion meshes but not so good on Morrowind meshes)
- Import: improved morrowind skeleton import (for example via base_anim files)

## 5.56 Version 2.1.15 (Oct 19, 2007)

- pycheck: added pychecker script (see http://pychecker.sourceforge.net/)
- test: added test script to automatically run importer and exporter on a range of selected nif and blend files
- Import/Export: PyFFI 0.5 is now required; the Blender scripts can now read and write a whole range of new nif versions (see PyFFI ChangeLog for details)
- Import/Export: small GUI improvements
- Import: ignore NiCamera root blocks instead of raising an exception on them
- Import: fixed a bug preventing animation import
- Import: fixed some progress bar issues
- Import: fixed bug in case armature parents another armature (i.e. solstheim's ice minion raider), this is still not working perfectly but at least the import completes without raising exceptions
- Import: `IMPORT_` prefix for realign option (in accordance with all other keys)
- Import: removed duplicate calculation of armature inverse matrix
- Import: replaced the deprecated method of linking armature to the scene
- Export: improved flatten skin so it works better in some cases

## 5.57 Version 2.1.14 (Oct 14, 2007)

- Import: fixed a transform bug which was introduced in 2.1.13, skinned geometries had their transform applied twice, so this fixes import of those skinned models that do not have a unit transform.
- Export: fixed a typo
- Import/Export/Config/GUI: restructured the scripts, in particular the import script has been transformed into an OOP class, so it requires no more globals for various settings. All gui and config related things have moved to a new nif_common.py library, as well as some common settings such as checking for Blender and PyFFI version. The result is that the code has been substantially simplified. The import and export script now also use exactly the same system to run the config gui.

## 5.58 Version 2.1.13 (Oct 13, 2007)

- Import: fixed transform error while joining geometries (this mostly affects the import of collision geometries)
- Import: optimized morph import yielding less array lookups and faster code
- Import: simplified texture searching and better linux support by looking for lower case versions of names too
- Import: automatically remove duplicate vertices after joining Morrowind collision geometries

## 5.59 Version 2.1.12 (Oct 11, 2007)

- Import: provide sensible error message on kf import
- Export: set flags to 0x000E for Oblivion ninodes and nitrishapes/nitristrips

- Export: automatically set blender collision type, draw type, and draw mode on old style (RootCollisionNode named mesh) morrowind collision export

## 5.60 Version 2.1.11 (Oct 3, 2007)

- Export: complain on unweighted vertices and select them, instead of adding an extra bone (this is a better alternative to the Scene Root.00 "feature" which was pretty frustrating at times when you had to hunt down unweighted vertices)
- Export: switched to using Mesh instead of using the deprecated NMesh
- Export: fixed frame time bug
- Import: removing dummy index does not properly delete the vertex from the mesh (yielding errors in the vertex key data), so reverted back to shift checking algorithm to fix face index order; the vertex order is shifted in place yielding simpler code and faster performance
- Import: removed _bindMatrix zombies, other minor cleanups
- Config: check blender version and raise exception if blender is outdated

## 5.61 Version 2.1.10 (Sep 27, 2007)

- Export: fairly large restructuring of the code, the Python modules are only loaded once
- Export: fixed alpha controller export
- Export: removed disfunctional material color controller export
- Export: added a timer
- Export: new option to merge seams between objects, if you separated meshes in different parts then on export often seams could appear between the parts (the better bodies meshes are good examples of this problem), now there is an option to recalculate the normals on seams between objects on export (for better bodies the result is a seamless body on re-export)

## 5.62 Version 2.1.9 (Sep 21, 2007)

- Export: new option to force dds extension of texture paths
- updated hull script for quickly creating bounding spheres

## 5.63 Version 2.1.8 (Sep 17, 2007)

- Export: new padbones option which pads and sorts bones as required by Freedom Force vs. The 3rd Reich
- Export: automatic settings for Freedom Force vs. The 3rd Reich
- Export: compacter gui
- new script for quickly creating bounding boxes

## 5.64 Version 2.1.7 (Sep 9, 2007)

- Import: trishapes/tristrips of grouping NiNodes are merged on import and the resulting merged mesh is named after the grouping NiNode
- Import: 'Tri ' prefix is no longer removed from name
- Import: simplified uv import and vertex color import code
- Import: fix for import of nifs with trishape/tristrip root
- Export: simplified heuristic for naming blocks
- Export: raise exception if bone names are not unique
- Export: fixed exception when bone name or armature name was very long
- Import/Export: support for Morrowind collision shapes using a polyheder bounds shape

## 5.65 Version 2.1.6 (Sep 5, 2007)

- Import: morrowind - better skeleton only import for better bodies
- Import: morrowind - better import for better bodies
- Export: make 'Bip01' root node also root of nif tree

## 5.66 Version 2.1.5 (Sep 2, 2007)

- Export: mopps for packed shapes
- Export: always strip texture paths (except for Morrowind and Oblivion)
- Import: shared texture folder detection for CivIV
- Import: assume stub has alpha channel if texture was not found and alpha property is present; this will ensure that NiAlphaProperty is written back on export

## 5.67 Version 2.1.4 (Aug 29, 2007)

- Export: fixed more bugs in bhkConvexVerticesShape
- Export: NiVertexColorProperty and NiZBufferProperty blocks for CivIV

## 5.68 Version 2.1.3 (Aug 19, 2007)

- Installer: also check in HKCU for registry keys of Python and PyFFI (fixes rare installation issue, see bug #1775859 on the SF tracker)
- new script for reducing number of influences per vertex, running this script before export helps if the skin partitioning algorithm complains about losing weights

## 5.69 Version 2.1.2 (Aug 17, 2007)

- Installer: make sure user is admin ("fixes" the Vista bug)
- Import: parent selected objects to armature when importing skeleton only
- Import/Export: Python profiler support (read Defaults.py for details)

## 5.70 Version 2.1.1 (Aug 14, 2007)

- Installer: open download page if dependency not found
- Export: make 'Scene Root' node scene root
- Export: quite a few bug fixes in Oblivion collision export, saner settings
- Export: option to toggle the use of bhkListShape
- Import: fix for skeleton.nif files
- Import: reverted to 2.0.5 bone import system if bone alignment is turned off, looks much better for Oblivion imports

## 5.71 Version 2.1 (Aug 12, 2007)

- Export: added support for Oblivion collisions
  - bhkBoxShape (from Blender 'Box' bounding shape)
  - bhkSphereShape (from Blender 'Sphere' bounding shape)
  - bhkCapsuleShape (from Blender 'Cylinder' bounding shape)
  - bhkPackedNiTriStripsShape (from Blender 'Static TriangleMesh' bounding shape)
  - bhkConvexVerticesShape (from Blender 'Convex Hull Polytope' bounding shape); Note that many of the settings are not well understood, so you probably still have to tweak the collision settings in nifskope. But at least the collision geometries should be properly exported.
- Export: fixed another bind position transform bug (reported by Corvus)
- Export: fixed a few other minor bugs

## 5.72 Version 2.0.7 (Aug 8, 2007)

- Import: added support for multiple skeleton roots
- Import: better support for meshes/armatures parented to bones
- Import: added option to send bones to bind position
- Import: added option to control application of skin deform
- Export: added option for stripification and strip stitching
- Export: fixed issue with non-uniform scaling on Freedom Force vs. 3rd Reich nifs
- Export: fixed issue with skin partition creation on older nif versions (such as Freedom Force vs. 3rd Reich nifs)

- Export: fixed problem with meshes sharing the same vcol lighting enabled material but not all having vertex weights (such as the Oblivion steel cuirass); the exporter now issues a warning rather than throwing an exception
- Export: fixed skin bounds calculation

## 5.73 Version 2.0.6 (Aug 6, 2007)

- Import/Export: fixed various transform errors
- Import: frames/sec detection
- Import: new and more reliable skinning import method
- Export: new options to control export of skin partition

## 5.74 Version 2.0.5 (Jul 30, 2007)

- Import: new option to import skeleton only
- Export: new options to export animation
- Export: 10.2.0.0-style transform controllers (includes Oblivion)
- Export: Morrowind style .kf files
- Export: fixed morph controller and morph data export
- Export: fixed getTransform on Zoo Tycoon 2 creatures

## 5.75 Version 2.0.4 (Jul 23, 2007)

- Import: fixed a few skin import transform errors (morrowind better bodies, oblivion armor)

## 5.76 Version 2.0.3 (Jul 22, 2007)

- Export: fixed skin export in case some bones did not influence any vertices
- Export: fixed transform error in skinned meshes such as better bodies and oblivion skeleton
- Export: support for 20.3.0.3 and 20.3.0.6 (experimental)

## 5.77 Version 2.0.2 (Jul 16, 2007)

- Import/Export: fix for config problem if nifscripts.cfg did not exist yet

## 5.78 Version 2.0.1 (Jul 14, 2007)

- Import: fix in transform of some skinned meshes
- Import/Export: simple local install script in .zip for linux

## 5.79 Version 2.0 (Jul 12, 2007)

- Import/Export: switched to PyFFI, support for NIF versions up to 20.1.0.3
- Import/Export: GUI revamped
- Export: tangent space calculation
- Export: skin partition calculation
- Export: skin data bounding sphere calculation
- Export: flattening skin hierarchy for oblivion

## 5.80 Version 1.5.7 (Jul 13, 2006)

- Import: further fix on zero length bones.
- Export: fixed export of unnamed objects.
- Export: fixed export of meshes parented to other meshes.

## 5.81 Version 1.5.6 (Jun 19, 2006)

- Export: fixed export of multi-material meshes.
- Export: fixed export of zero-weighted vertexes.

## 5.82 Version 1.5.5 (Jun 15, 2006)

- Import: fixed import of zero length bones.
- Export: fixed export of meshes with no parents.

## 5.83 Version 1.5.4 (Jun 12, 2006)

- Export: fixed a bug in apply_scale_tree

## 5.84 Version 1.5.3 (Jun 10, 2006)

- Export: fixed an issue with skinned models (clothing slots now no longer require to be applied transformation with NifSkope)
- Import: fixed import of animation keys
- Export: no more empty NiNode at the end of bone chains
- Export: optimized the export of single materialed, non-animated meshes.
- Import/Export: bone names are restored

## 5.85 Version 1.5.2 (Apr 19, 2006)

- Export: new option APPLY_SCALE (on by default) which resolves TESCS selection box issue and a 1.5 incompatibility problem
- Import/Export: full Python installation no longer needed
- Export: keyframe data realigned as well (should allow us, in theory, to re-export base animation files)
- Export: transform fix on dummy tail NiNodes
- Import: if texture not found, a stub is created
- Export: bone optimization fix
- Import: realignment is now always automatic
- Import/Export: correction on 1.5.1 ChangeLog, you'll still need the Bip01 spell, but we're getting closer

## 5.86 Version 1.5.1 (Apr 13, 2006)

- Export: a 20.0.0.4 bug is fixed
- Import/Export: restoring bone matrices, no longer need for NifSkope's Bip01 spell
- Import: animated nodes that aren't bones have their animation imported too
- Import/Export: scaling fix
- Import: initial attempt to use the original NIF bone matrices if auto-align is turned off

## 5.87 Version 1.5 (Mar 21, 2006)

- Import: fix for models that have a NiTriShape as root block
- Import: added config option to retain bone matrices
- Import: full animation support, animation groups and keyframes
- Import: detects invalid / unsupported NIF files
- Export: bugfix in animation export
- Export: bugfix in vertex weight export
- Export: large model fix (now supports up to 65535 faces / vertices per mesh
- Export: writes a dummy node on final bones to retain bone length when re-imported

## 5.88 Version 1.4 (Feb 12, 2006)

- Import: completely rewritten, uses Niflib now just like exporter
- Import/Export: support for all NIF versions up to 20.0.0.4!!
- Import/Export: corrected specularity import/export (thanks NeOmega)
- Import/Export: hidden flag via object wire drawtype

- Import: full skinning support (but still no animation)
- Import: better bone length estimation, automatic alignment

## 5.89 Version 1.3 (Jan 21, 2006)

- Import/Export: Vertex key animation support (geometry morphing).
- Export: Bugfix in bone animation export (transformations sometimes wouldn't show up correctly before).
- Import: Improved bone length calculation.
- Export: Added NIF v10.0.1.0 support.
- Export: Skinning bugfix for multimaterialed meshes.
- Export: Vertex weight calculation optimized, and no more annoying console messages!
- Export: Embedded textures reestablished.

## 5.90 Version 1.2 (Dec 23, 2005)

- Import/Export: updated for Blender 2.40
- Export: now uses Niflib, which implies that it runs much faster, the code is much cleaner, and multiple NIF version support is in the making
- Export: replaced old crappy config file system with Blender's native Script Config Editor system
- Export: new feature - texture flipping
- Export: new feature - export of bones, armatures, and vertex weights (finally!!!)
- Export: packed texture feature has been temporarily dropped; this functionality is being transferred to Niflib

## 5.91 Version 1.1 (Oct 31, 2005)

- Export: Fixed bug pointed out by Sabregirl, on mesh_mat_shininess.
- Export: Applied m4444x's patches to exporter (texture flipping), changed names, included exporter readme file.
- Import/Export: Changed the licensing to BSD.
- Import: Added support for texturing in the editor 3D view. Now the textures will show up in textured mode if loaded.
- Import: NiMorph Controllers that m4444x coded. Haven't tested it, but it doesn't break the previous functionality, so it should be fine
- Export: Added an option for stripping the texture's file path
- Export: Support for subsurfed meshes (display level).
- Export: Vertex export method improved, extreme speedup!
- Import/Export: Transparency support improved.
- Import: Small fix in the import of vertex colors.

- Import: Autodetect Morrowind style texture path; if you load a NIF from ...meshes... then the importer will look in ...textures* for the NIF textures.
- Export: Fixed animation group export.
- Import: Multiple texture folders.
- Import/Export: number of vertices and number of faces is unsigned short: fix in importer, and added range check in exporter.
- Import/Export: Added glow mapping.
- Export: Fixed texture flipping
- Import/Export: Config file support.
- Import/Export: Now we have a GUI for setting various options.
- Import: Solved problem with textures embedded in NIF file; textures will not load but the script will still load the meshes.

## 5.92 Version 1.0 (Oct 12, 2005)

- Initial bundled release of the importer v1.0.6 and exporter v0.8 on SourceForge.

CHAPTER 6

# License

CHAPTER 7

# Indices and tables

- genindex
- modindex
- search