mirai
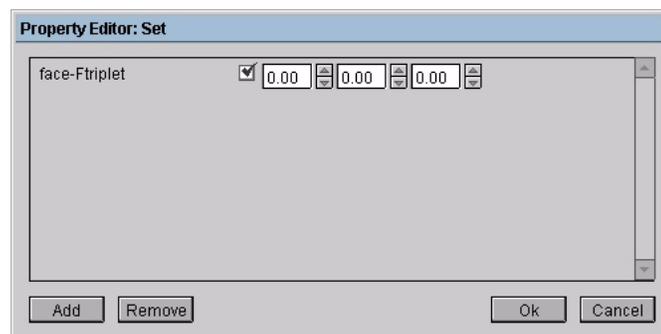
USING THE PROPERTY EDITOR PLUG-IN

This tutorial will teach you how to create and assign custom properties for export to a target platform using the Property Editor plug-in.

ESTIMATED TIME REQUIRED

20 Minutes

LEARNING GOALS

In this tutorial, you will learn about:

- Defining custom properties

- Assigning custom properties in Mirai

- Using the Property Editor

- Exporting custom properties

- Converting N-World 3.2 Properties for use in Mirai

- Setting Property Editor preferences

ABOUT CUSTOM PROPERTIES

In addition to the standard material and texture properties assigned to an object (such as diffuse color or vertex color), you can tag different geometric elements with custom properties that your target platform understands. For example, if your target platform understands a property you've defined as "infrared," you can easily assign a vertex, edge, face, polyhedron, joint, bone, or skeleton the "infrared" property with some arbitrary value, then export that data through Nichimen's Game Exchange data format. Properties are assigned through the Property Editor, which is available when the Property Editor plug-in is loaded and active.

CREATING CUSTOM PROPERTIES

You define the properties that will appear in the Property Editor by creating a text file containing the property definitions. You can download a sample file, example-property-file.npf with this tutorial. The file defines a set of properties that can be assigned to vertices, edges, faces, polyhedra, joints, bones, or skeletons in Mirai.

The example-property-file.npf file can be divided into two sections: a description of the contents of the property file and the property definitions.

The lines in the property file that begin with semicolons are comments which describe how the property file works. These lines are not required to create or define a property, they are included as an explanation of the properties in the file. A portion of this documentation section is shown below.

```
;;; There is an example property file below. When you are defining a property file
;;; of this form, you are defining a lisp form file that is loaded in conjunction with
;;; the property editor. The interface used for this is in the "load property template"
;;; menu element in the plugins menu. After being loaded once the properties defined
;;; will persist in the scene without any requirement for reloading. However, you
;;; should be aware that this is a lisp file that you are loading, therefore it is
;;; very important that you adhere to the syntax of the file.
;;;
;;; The code is as follows:
;;;
;;;    (bone      :mass-bone  "Mass on bone"   float           0.0   "Doc string")
;;;    (joint     :i-mass-joint    "i-Mass joint"   integer          0     "Doc string")
;;;    ;;
;;;    ;; this list can be any length
;;;    ;;
;;;    (polyhedron     :polyhedron-col  "polyhedron-color"    rgb-color   )
:::
```

The property definitions are listed after the description section. In this section, the properties and their default values are defined.

```
(bone          :mass-bone  "Mass on bone"   float          0.0   "Doc string")
(bone          :i-mass-bone     "i-Mass bone"    integer        0     "Doc string")
(joint         :i-mass-joint    "i-Mass joint"   integer        0     "Doc string")
(skeleton  :i-mass-skel     "i-Mass skel"    integer        0    "Doc string")
(vertex    :mass-vertex     "mass vertex"    float          0.0   "Doc string")
(edge          :edge-string   "edge-string" string         "nothing")
(edge          :edge-mass2 "edge-mass2"      float          0.0   "Doc string")
(edge          :edge-fdoublet "edge-Fdoublet"   fdoublet    (1.0 1.0))
(edge          :edge-ftriplet "edge-Ftriplet"   ftriplet    (1.0 1.0 1.0))
(edge          :edge-fquad     "edge-Fquad"  fquadruplet (1.0 1.0 1.0 1.0))
(face          :face-ftriplet "face-Ftriplet"   ftriplet    )
(face          :face-mass3 "face-mass3"      float          0.0   "Doc string")
(polyhedron     :polyhedron-mark  "polyhedron-mark?"      boolean     nil      "Doc string")
(polyhedron     :polyhedron-mark-t     "polyhedron-mark-t?"   boolean     t    "Doc string")
(polyhedron     :polyhedron-col  "polyhedron-color"      rgb-color   (0.5 0.5 0.5))
```

Property definitions must be formatted as follows in order for them to appear in the Property Editor:

```
(element :keyword "Label" type default "Documentation")
```

- *Element* defines what type of element this property affects. An element can be any of the following:

  - Vertex

  - Edge

  - Face

  - Polyhedron

  - Joint

  - Bone

  - Skeleton

- *Keyword* is the name that appears when the scene containing the custom properties is exported. Keywords must be preceded by a colon and must be unique for that element type. For example, the keyword in the first property listed above is :mass-bone. If you wanted, you could add another property named :mass-bone to the file, but it would have to be assigned to a different element type.

NOTE: *You cannot use the following characters for a keyword: ~&*%$#@?!'\".*

- *Label* is the name that appears in the Property Editor. It must be enclosed within quotes and it also must be unique for that element type. For the mass-bone property, "Mass on bone" is the label that will appear in Mirai's property editor.

- *Type* is the type of value that can be used with this property. You can use any of the following:

  - *Float* is a floating point number.

  - *Integer* is a whole number.

  - *Fdoublet* is a group of two floating point numbers.

  - *Ftriplet* is a group of three floating point numbers.

  - *Fquadruplet* is a group of four floating point numbers.

  - *String* is a non-numerical value.

  - *Boolean* is a boolean value. It can be either true (t) or false (nil).

- *RGB-color* is a color defined using RGB values.

- *Default* is the value that will initially appear in the Property Editor. This is optional when defining a property and can be any value that is appropriate for that type. If a value is not specified, then the default value shown below is used.

TYPE	DEFAULT VALUE	EXAMPLE VALUE
Float	0.0	5.0
Integer	0	7
Fdoublet	(0.0 0.0)	(2.0 2.0)
Ftriplet	(0.0 0.0 0.0)	(4.0 4.0 4.0)
Fquadruplet	(0.0 0.0 0.0 0.0)	(6.0 6.0 6.0 6.0)
String	""	"Name"
Boolean	nil	t
RGB color	(0.0 0.0 0.0)	(1.0 0.0 0.0)

- *Documentation* is a string that provides text for the information bar in Mirai. This is optional.

When all of the properties are defined, you need to save the file as an .NPF file so that it can be read by the Property Editor. It can be saved in any location, just note the location for future reference. Once saved, these properties can be assigned to objects in Mirai.

USING THE PROPERTY EDITOR PLUG-IN

The Property Editor plug-in assigns and removes properties to geometric elements and performs searches based on property assignments. Follow the steps below to get some practice using the Property Editor.

ASSIGNING PROPERTIES TO OBJECTS

You assign properties to objects using the Property Editor. Once assigned, properties will stay assigned to the object unless you remove them.

NOTE: *If you duplicate an object that has properties assigned to it, the properties will not be assigned to the copy.*

● TO ASSIGN PROPERTIES TO OBJECTS IN MIRAI:

1 [L] on *Applications>Plug-In Manager* to start the Plug-In Manager. The Plug-in Manager controls which plug-ins are loaded and active.
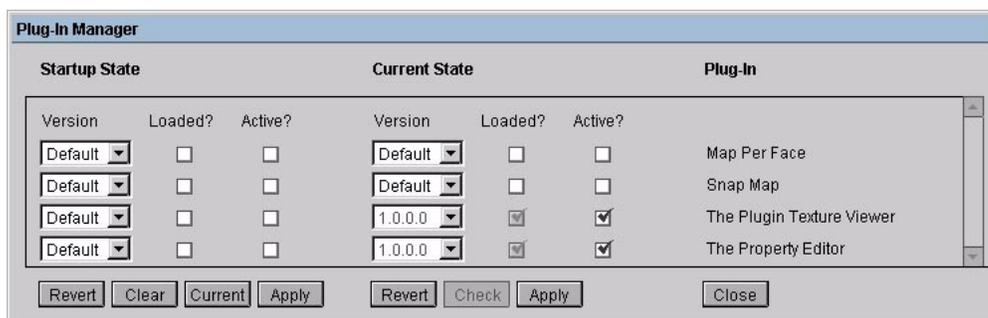
- The *Startup State* columns show the status of any plug-ins at start-up.

- The *Current State* columns show the current status of any plug-ins.

- *Revert* returns all settings to the state they were in when the Plug-In Manager was opened.

- *Clear* undoes any changes you have made in the Current State column.

- *Current* copies the Startup State settings to the Current State column.

- *Check* verifies that the selected combination of plug-ins is valid. A combination of plug-ins is invalid if the plug-ins use different versions of the same plug-in or require an unloaded plug-in to run. If the plug-in combination is invalid, an error message appears.

- *Apply* accepts any changes you have made in the Plug-In Manager.

- *Close* closes the Plug In Manager.

2 [L] in the *Active* checkbox in the Current State column next to the Property Editor. This loads and activates the Property Editor when this setting is applied.

NOTE: *If you [L] in the Loaded checkbox, the plug-in is loaded, but not activated.*

3 [L] on *Apply*. The Property Editor plug-in is loaded and activated.

4 [L] on *Close* to close the Property Editor.

5 [L] on *Applications>Plug-Ins>Load Property Template*. Load Property Template lets you load the file containing the properties you wish to assign to objects.

NOTE: *You can have multiple property files loaded simultaneously.*
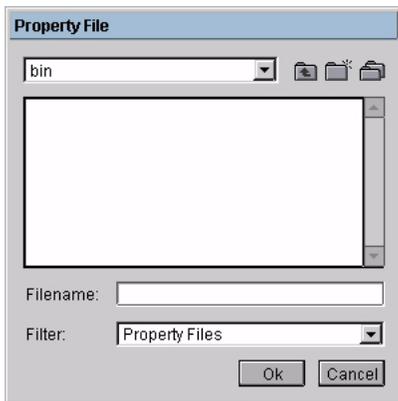
6 In the Property File dialog box, navigate to the example-property-file.npf file that you downloaded with this tutorial, then [L] on *Ok* to load the file. The properties are made available in the scene.

NOTE: *If any of the property definitions in a file contain improper syntax, the property will not be added to the scene.*

7 Create a cube and make it visible in the Geometry editor.

8 Select a face on the cube, then [R].

9   In the menu that appears, [L] on *Set Property*. The Set Property command assigns custom properties to elements using the Property Editor. You assign and select properties using the Property Editor.


- *Add* adds properties to the Property Editor dialog box so that you can edit their values. The add button does not assign the properties to the selected element, it only makes them available for editing in the Property Editor.

- *Remove* removes properties from the Property Editor dialog box. Properties are not removed from the elements to which they have been assigned, they are only removed from the Property Editor.

- *Ok* assigns the properties to the selected elements and closes the Property Editor.

- *Cancel* closes the Property Editor without assigning properties to the selected elements.

NOTE: *You can only have one Property Editor open at a time.*

10   [L] on the *Add* button in the Property Editor. A dialog box containing the properties that can be assigned to faces appears.

11   Select *face-Ftriplet* from the dialog box, then [L] on *Ok*. The face-Ftriplet property specifies an Ftriplet to be assigned to the face. The face-Ftriplet property appears in the window.
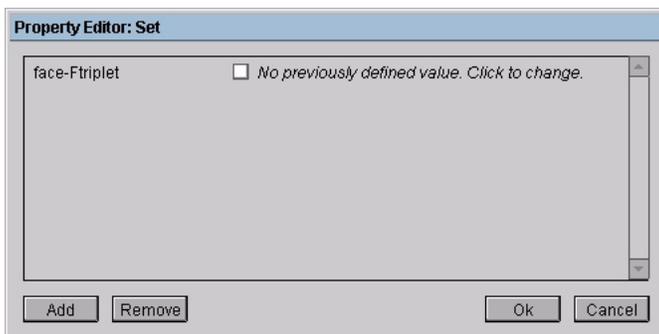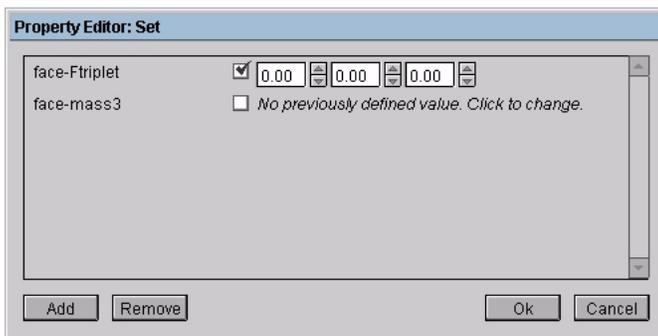
12 [L] in the checkbox next to face-Ftriplet to define the property's value for the selected face. When checked, the value specified in the example-property-file.npf file appears.
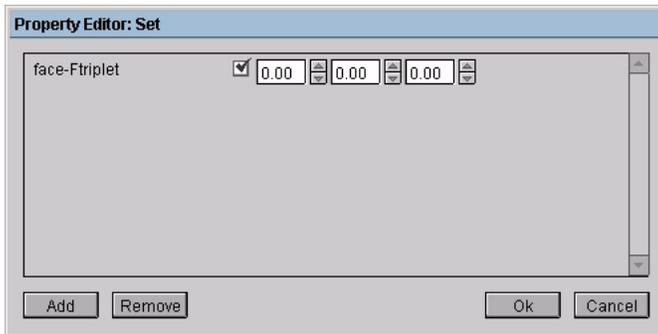

13 Set face-Ftriplet to 1.0 1.0 1.0.

14 [L] on *Add*, then select *face-mass3* from the Property Editor dialog box and [L] on *Ok*. face-mass3 appears in the Property Editor window.

15 [L] in the *face-mass3* checkbox to define face-mass3's value for the selected face. When checked, the default value for face-mass3 appears.

16 [L] in the *face-mass3* field to set its value to 5, then [L] on *Ok*. The face-mass3 and face-Ftriplet properties are assigned to the selected face using the specified values.

17 [R] on the same face, then [L] on *Clear Property* in the menu that appears. Clear Property allows you to remove property assignments from the selected elements.

The Clear Property dialog box appears. It lists all of the properties assigned to that face, so that you can select the one that you would like to remove.
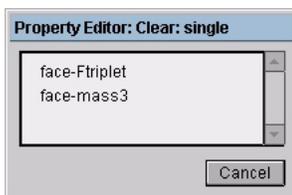

NOTE: *To remove more than one property from an element, [R] on Clear Property, then Ctrl L on each property you want to delete.*

18 [L] on *face-Ftriplet*. The property is no longer assigned to the face.
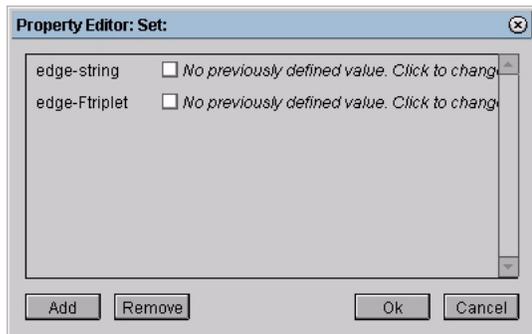
19 Select three of the cube's edges.
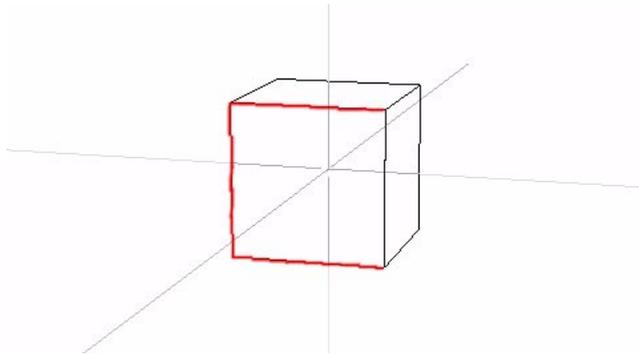
20 [R] on the collection, then [L] on *Set Property>Add*. The Property Editor dialog box appears. This time, it contains the properties that can be assigned to an edge.
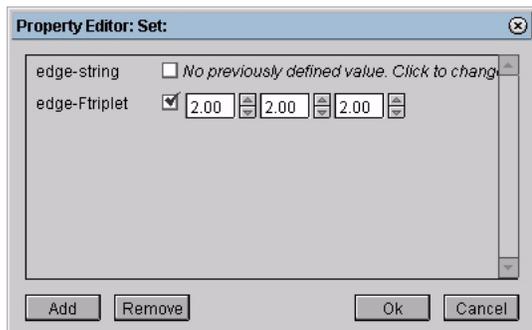
Add Property:

```
edge-string
edge-mass2
edge-Fdoublet
edge-Ftriplet
edge-Fquad
```

Ok   Cancel
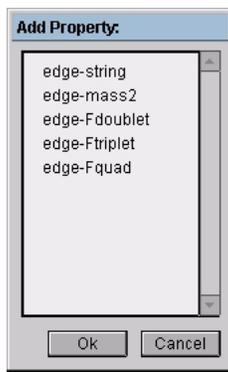
21 Ctrl [L] on *edge-Ftriplet* and *edge-string*, then [L] on *Ok*. Both properties appear in the Property Editor.

Property Editor: Set:

| edge-string | ☐ No previously defined value. Click to change |
| edge-Ftriplet | ☐ No previously defined value. Click to change |

Add   Remove   Ok   Cancel

22 [L] in the checkbox next to edge-Ftriplet to define its value for the edges you selected.

23 Set edge-Ftriplet to 2.0 2.0 2.0. This value will be assigned to each of the edges you selected.
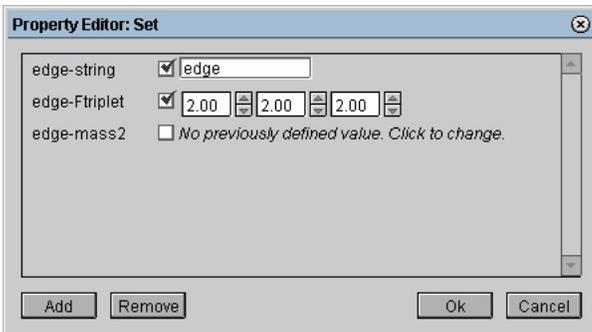
Property Editor: Set:

| edge-string | ☐ No previously defined value. Click to change |
| edge-Ftriplet | ☑ 2.00 ⬍ 2.00 ⬍ 2.00 ⬍ |

Add   Remove   Ok   Cancel

24 [L] in the *edge-string* checkbox to define its value, then [L] in the *edge-string* field and enter the string "Edge."
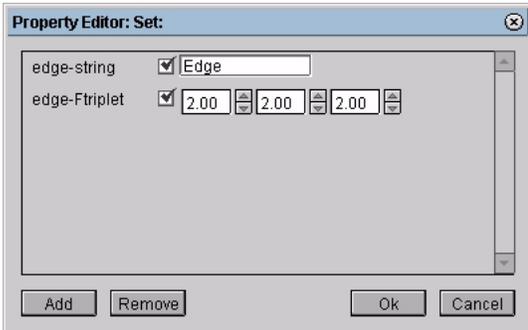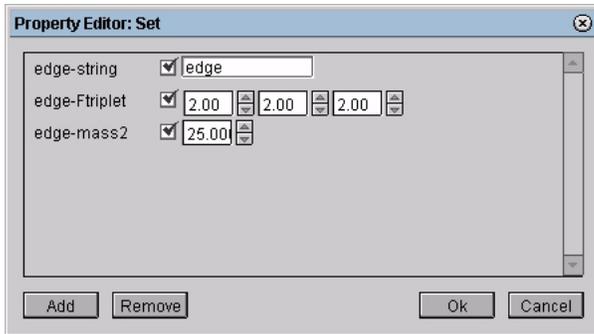

25 [L] on *Ok*. The values are assigned to the selected edges.

26 Select a different edge on the cube.

27 [R] on that edge, then [M] on *Set Property*. When you [M] on Set Property, the last values assigned to that element type appear as the default value.
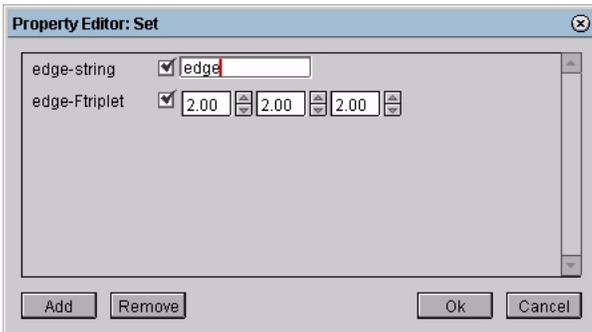
28 [L] on *Add* to add a property to the Property Editor. The Property Editor dialog box appears.

29 [L] on *edge-mass2*, then [L] on *Ok*. Edge-mass2 is added to the Property Editor window.

30 [L] on the checkbox next to edge-mass2, then specify a value of 25 for it.

31 [L] on *Ok*. All three values are assigned to the selected edge.

# SELECTING ELEMENTS BASED ON PROPERTY ASSIGNMENTS

In addition to assigning properties with the Property Editor, you can also use the Property Editor to search for points, segments, faces, joints, or bones that have a certain property assigned to them.

32 [R] on the cube's body, then [M] on *Select on Property* in the menu that appears. The Select on Property command lets you select elements within the selected area of the object or objects based on their current property assignments.

   • [L] highlights any *points* within the selected portion of the object that have the chosen property assigned to them. If you have a portion of a skeleton selected, [L] to highlight joints that have the chose property assigned to them.

   • [M] highlights any *segments* within the selected portion of the object that have the chosen property assigned to them. If you have a portion of a skeleton selected, [M] to highlight bones that have the chose property assigned to them.

   • [R] highlights any *faces* within the selected portion of the object that have the chosen property assigned to them
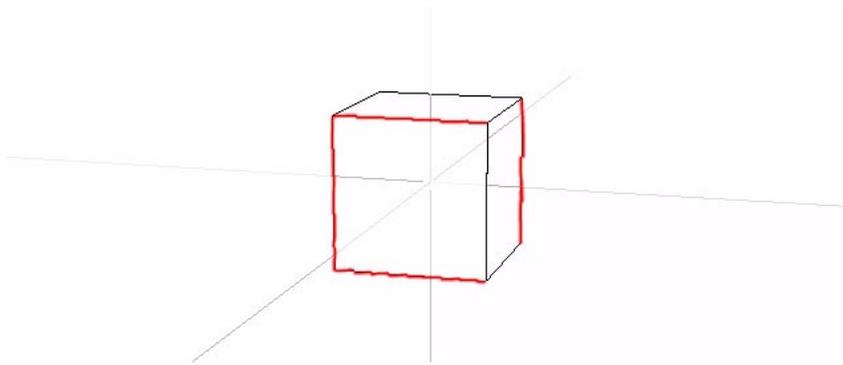
33 [L] on *Add* to open the Property Editor Select dialog box.

34 In the dialog box, [L] on *edge-Ftriplet*, then [L] on *Ok*. edge-Ftriplet is added to the Property Editor window.
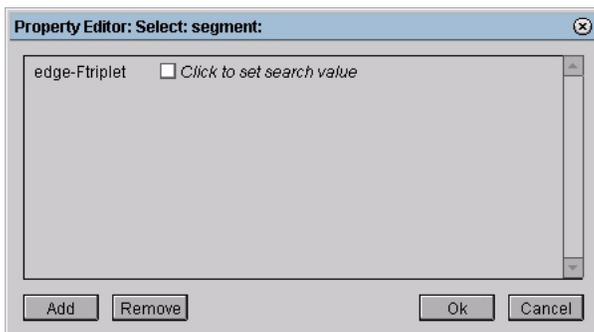
**35** [L] in the *edge-Ftriplet* checkbox, then set the search value to 2.0 2.0 2.0.

36 [L] on *Ok*. All of the edges on the cube with that value are highlighted in the Geometry window.


SELECTING PROPERTIES ON MULTIPLE POLYHEDRA

You can select properties on multiple polyhedra using the *Select on Polyhedron* command.

37 Add a sphere to the Geometry window and make it visible.

38 [R] on the sphere's body, then [L] on *Set Property*.

39 [L] on *Add*, then select polyhedron-color from the dialog box that appears and [L] on Ok. The polyhedron-color property is added to the Property Editor.

40 [L] in the polyhedron-color field to bring up the Color Editor.

41 Make red the current color in the Color Editor, then [M] in the polyhedron-color field to make that red the value for polyhedron-color, then [L] on *Ok*.
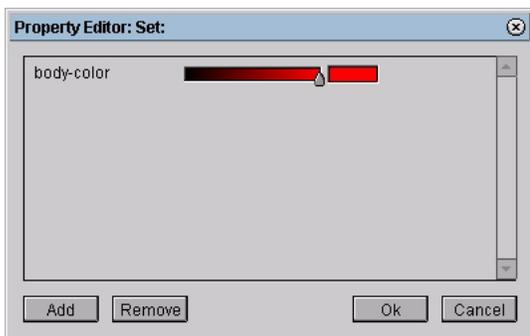

**42** In the Color Editor, [M] in an empty scratch pad slot to store the color for later use.
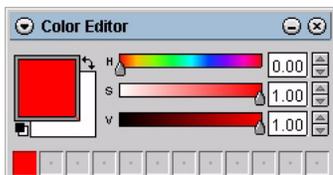
43 Select the cube's body, then [R].

44 In the menu that appears, [R] on *Set Property*. This assigns the polyhedron-color property value that you just assigned to the sphere's body to the cube's body.

45 Select the sphere and cube's bodies, then [R].

46 In the menu that appears, [L] on *Select on Polyhedron*. This command allows you to find the polyhedron to which a certain property has been assigned from amongst the selected polyhedra.

47 [L] on Add, then select *polyhedron-color* from the dialog box, then [L] on *Ok* to search on the polyhedron-color property.

Add Property:

body-mark?
body-mark-t?
body-color
body-filename

Ok   Cancel
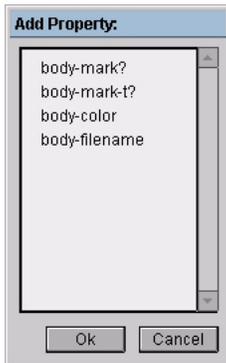
48 In the Property Editor, [L] on the *polyhedron-color* checkbox, then use the red that you assigned to the polyhedra bodies earlier as the search value. To make that same shade of red the search value:

- [L] on the red value you saved to make it the current color in the Color Editor

- [M] in polyhedron-color field to make red the search value.

Property Editor: Set:

body-color

Add   Remove   Ok   Cancel
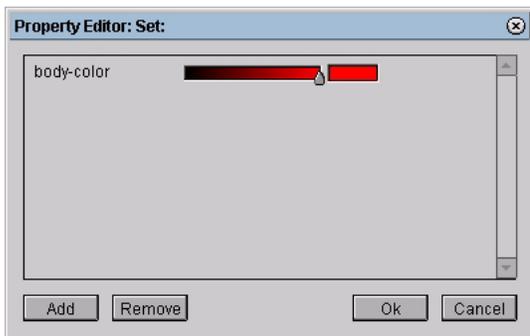
49 [L] on *Ok*. The sphere and the cube are highlighted.

# SELECTING PROPERTIES ON MULTIPLE SKELETONS

You select properties on multiple skeletons using the *Select on Skeleton* command.

50 Add a skeleton to the scene.

51 [R] on the skeleton's body, then [L] on Set Property.

52 [L] on *Add*, then select i-Mass skel from the dialog box that appears and [L] on Ok. i-Mass skel is added to the Property Editor.
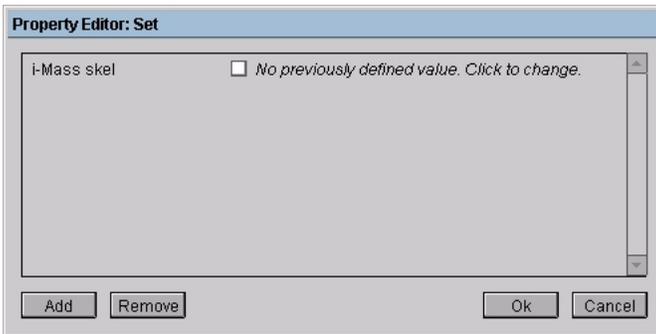
53 [L] in i-Mass' checkbox to reveal its value.

54 Set i-Mass to 3, then [L] on Ok. The i-Mass property is assigned to the selected bone.

55 Add another skeleton to the scene.

56 Select both of the skeletons in the scene, then [R].

57 In the menu that appears, [L] on Select on Skeleton. The Property Editor appears.
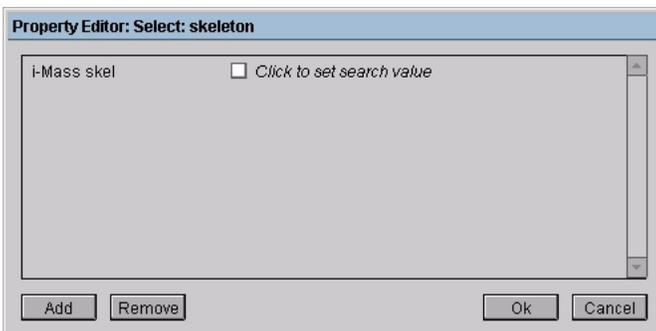
58 [L] in the checkbox next to i-Mass skel to set the search value.

59 Set the search value to 3, then [L] on Ok. The skeleton with the i-Mass skel property assigned to it is highlighted, the other skeleton is not.

REMOVING A PROPERTY FILE

60 [L] on *Applications>Plug-Ins>Remove Property Template* to make the properties unavailable in the scene. The Remove Template dialog box appears.
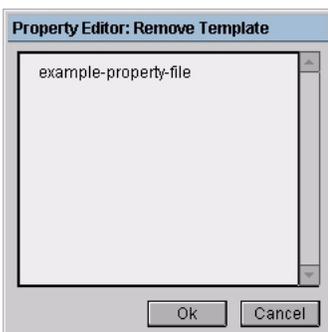
61 [L] on *example-property-file*, then [L] on *Ok*. The property file is removed from the scene, but all of the property assignments you made earlier are still valid.

EXPORTING CUSTOM PROPERTIES

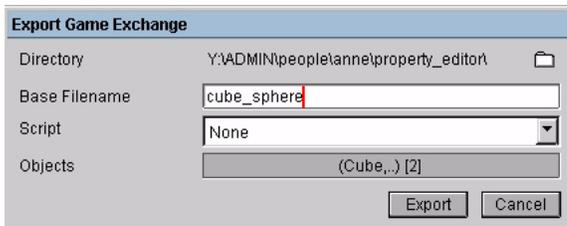Once properties are assigned in Mirai, they can be written out by exporting to Game Exchange. The information contained in these files can then be accessed directly when converting those files to the target format.

To export to Game Exchange:

62 [L] on *File>Export>Game Exchange* to export the current scene to Game Exchange. The Export Game Exchange dialog box appears.

63 Select the directory to which you want to export the scene.

64 Enter cube_sphere for the base filename. The base filename is the name of the scene when it is exported.

65 [L] in the *Objects* field and select all of the objects in the scene from the dialog box that appears. These are the objects that will be exported with the scene.

66 [L] on *Export*. The scene is exported to the directory you specified. When the export is complete, the Export Game Exchange dialog box appears.

The following files are exported to the directory you specified:

- Cube.gof
- cube_sphere.cam
- cube_sphere.gef
- cube_sphere.gmf
- cube_sphere.grp
- cube_sphere.lgt
- gaf-2-03.tpl
- gef-2-03.tpl
- gmf-2-03.tpl

- gof-2-03.tpl

- gsf-2-03.tpl

- Human_Skeleton.gsf

- Human_Skeleton_2.gsf

- Human_Skin.gof

- Human_Skin_2.gof

- Sphere.gof

67 Using a text editor, open the Cube.gof file you just created. GOF files describe a single object, including any properties that are assigned to it.

```
body Polyhedron-67 (
  vertices[] <
    (coord  -10.0 -10.0 10.0;)
    (coord  -10.0 10.0 10.0;)
    (coord  10.0 10.0 10.0;)
    (coord  10.0 -10.0 10.0;)
    (coord  10.0 -10.0 -10.0;)
    (coord  10.0 10.0 -10.0;)
    (coord  -10.0 10.0 -10.0;)
    (coord  -10.0 -10.0 -10.0;)>
  faces[] <
    (normal 0.0 0.0 1.0;
     vertex-indices[] < 0; 1; 2; 3;>
     vertex-normal-indices[] < 0; 1; 2; 3;>)
    (normal 0.0 0.0 -1.0;
     vertex-indices[] < 4; 5; 6; 7;>
     vertex-normal-indices[] < 4; 5; 6; 7;>)
    (normal 0.0 1.0 0.0;
     vertex-indices[] < 1; 6; 5; 2;>
     vertex-normal-indices[] < 1; 6; 5; 2;>)
    (normal 0.0 -1.0 0.0;
     vertex-indices[] < 7; 0; 3; 4;>
     vertex-normal-indices[] < 7; 0; 3; 4;>)
    (normal 1.0 0.0 0.0;
     vertex-indices[] < 3; 2; 5; 4;>
     vertex-normal-indices[] < 3; 2; 5; 4;>
     properties (
        FACE-MASS3 5.0;))
    (normal -1.0 0.0 0.0;
     vertex-indices[] < 7; 6; 1; 0;>
     vertex-normal-indices[] < 7; 6; 1; 0;>)>
```

If you open the Sphere. gof or the Human Skin.gof files, you can see the other properties that were assigned. For information on converting game exchange files, see the *Game Exchange 2.0 Programmer's Guide*.

CONVERTING N-WORLD 3.2 PROPERTY DEFINITIONS

Objects that had properties assigned to them in N-World 3.2 will still have the properties assigned to them when you import the object into Mirai, however, the properties will not be visible in the Property Editor. In order to edit those properties or assign them to other objects in the scene, you need to convert the lisp file containing the property definitions into an .NPF file so that the Mirai's Property Editor will recognize them.

You can download a sample N-World 3.2 property file, custom-properties.lisp with this plug-in. In the following section, you will convert it into an .NPF file so that the properties can be edited and assigned by Mirai's Property Editor.

To convert a lisp file into an .NPF file:

1 In a text editor, open the custom-properties.lisp file that you downloaded with this plug-in.

```
(in-package "GEOMETRY")

(make-simple-custom-template 'integer nil :default 0.0 :widget 'ui::number-field)

(make-compound-custom-template 'description '((integer IDnumber) (bool modified?) (string note)))

;; add property to property-editor

(add-custom-properties '(
                              (point "point-color" color)
                              (point "point-bool" bool)
                              (point "point-integer" integer)
                              (point "point-float" float)
                              (point "point-ftriplet" ftriplet)
                              (point "point-string" string)
```

2 Delete any line that does not define a property. This includes the add-custom-properties form and all of the lines the lines prior to it. These lines are not needed to define properties in Mirai.

```
(point "point-color" color)
(point "point-bool" bool)
(point "point-integer" integer)
(point "point-float" float)
(point "point-ftriplet" ftriplet)
(point "point-string" string)
```

3 Replace each property's Property Class with the appropriate Element, shown below. Element defines what type of element this property affects.

N-WORLD 3.2 PROPERTY CLASS	MIRAI 1.0 ELEMENT
point	vertex
not supported	edge
face	face
body	polyhedron
object	not supported
not supported	joint
not supported	bone
not supported	skeleton

NOTE: *You cannot assign properties to objects in the Property Editor in Mirai.*

For example, the property "point-color" was assigned to points in N-World 3.2. In Mirai, it will be assigned to vertices.

```
(vertex "point-color" color)
(vertex "point-bool" bool)
(vertex "point-integer" integer)
(vertex "point-float" float)
(vertex "point-ftriplet" ftriplet)
(vertex "point-string" string)
```

4 Create a keyword for each property. In N-World 3.2, the property's name identified the property when it was exported, so there is no equivalent parameter in N-World 3.2. For more information on creating keywords, see page 3.

```
(vertex :vertex-color "point-color" color)
(vertex :vertex-boolean "point-bool" bool)
(vertex :vertex-integer "point-integer" integer)
(vertex :vertex-float "point-float" float)
(vertex :vertex-ftriplet "point-ftriplet" ftriplet)
(vertex :vertex-string "point-string" string)
```

5   Create a label for each property. If desired, you can use the name from 3.2 as the property's label. For more information on creating labels, see page 3.

```
(vertex :vertex-color "point-color" color)
(vertex :vertex-boolean "point-bool" bool)
(vertex :vertex-integer "point-integer" integer)
(vertex :vertex-float "point-float" float)
(vertex :vertex-ftriplet "point-ftriplet" ftriplet)
(vertex :vertex-string "point-string" string)
```

6   Replace the property's Template Definition with the appropriate Type. Type is the type of value that can be used with this property. You can use any of the following:

N-WORLD 3.2 TEMPLATE DEFINITION	MIRAI 1.0 TYPES
float	float
ftriplet	ftriplet
string	string
bool	boolean
color	rgb-color
integer	integer
not supported	fdoublet
not supported	ftriplet
not supported	fquadruplet
simple custom template	not supported
compound custom template	not supported

NOTE: *You cannot use simple or compound custom templates with the Mirai version of the Property Editor. Simple and compound custom templates are described in the Game Exchange 2.0 Programmer's Guide.*

```
(vertex :vertex-color "vertex-color" rgb-color)
(vertex :vertex-boolean "point-bool" boolean)
(vertex :vertex-integer "point-integer" integer)
(vertex :vertex-float "point-float" float)
(vertex :vertex-ftriplet "point-ftriplet" ftriplet)
(vertex :vertex-string "point-string" string)
```

7   Specify default values for each property, if desired. For more information on default values, see page 4.

```
(vertex :vertex-color "vertex-color" rgb-color)
(vertex :vertex-boolean "point-bool" boolean nil)
(vertex :vertex-integer "point-integer" integer 7)
(vertex :vertex-float "point-float" float 14.0)
(vertex :vertex-ftriplet "point-ftriplet" ftriplet)
(vertex :vertex-string "point-string" string)
```

8   Create documentation strings for each property, if desired. For more information on documentation strings, see page 4.

```
(vertex :vertex-color "vertex-color" rgb-color "Doc string")
(vertex :vertex-boolean "point-bool" boolean nil)
(vertex :vertex-integer "point-integer" integer 7)
(vertex :vertex-float "point-float" float 14.0 "Doc String")
(vertex :vertex-ftriplet "point-ftriplet" ftriplet)
(vertex :vertex-string "point-string" string)
```

9   Save the file as an .NPF file. Unlike in N-World 3.2, you can save the property file under any name and in any location as long as it has an .NPF extension. The properties in this file can now be edited and assigned using the Property Editor in Mirai.

SETTING PROPERTY EDITOR PREFERENCES

To set the preferences that control how properties are displayed in the Property Editor, [L] on *File>Preferences>Property Editor>Item Display*. The preferences you can set for the Property Editor are revealed:

- *Show previously edited values* displays properties that have already been added to the Property Editor for the selected element type. So, once you assign properties to an element type, those properties will automatically appear in the Property Editor each time you open the Property Editor with that element type selected.

- *Show set values* displays any property values that have been previously set for the selected element type after the property's checkbox has been checked.

COMMAND AND MENU ITEM SUMMARY

[R] on a vertex, edge, face, polyhedron body, joint, bone, or skeleton or a collection of these elements to bring up a context-sensitive menu containing some or all of the following commands:

- *Set Property* assigns properties to elements using the Property Editor.
  - [L] to assign properties to the selected element using the Property Editor.
  - [M] to assign properties to the selected element using the Property Editor. When the Property Editor opens, the last properties and values assigned to that element type appear as defaults.
  - [R] to assign the last properties and values assigned to that element type to the selected element.

- *Clear Property* removes property assignments from the selected elements.
  - [L], [M] to remove a single property from selected elements.
  - [R] to remove more than one property from the selected element.

- *Select on Property* highlights elements in the current selection based on their property assignments. The selection can be within single or multiple polyhedra or within single or multiple skeletons.
  - [L] to highlight any points or joints within the current selection that have the chosen property assigned to them.
  - [M] to highlight any segments or bones within the current selection that have the chosen property assigned to them.
  - [R] to highlight any faces in the current selection that have the chosen property assigned to them.

- *Select on Polyhedron* highlights polyhedra in the current selection to which a property has been assigned.
  - [L], [M], [R] to select the polyhedron or polyhedra to which a property has been assigned.

- *Select on Skeleton* highlights skeletons in the current selection to which a property has been assigned.
  - [L], [M], [R] to select the skeleton or skeletons to which a property has been assigned.

[L] on *Applications>Plug-Ins* to access either of the following commands:

- *Load Property Template* makes properties available in a scene for assignment.

- *Remove Property Template* makes properties unavailable in a scene. Once removed, the properties can no longer be assigned to additional elements, but they are not removed from the elements to which they have already been assigned.