ComputerWorks

AUTHORIZED DISTRIBUTOR

# GRAFISCHES SCRIPTEN MIT MARIONETTE

## EINE EINFÜHRUNG

VECTORWORKS®

**DISTRIBUTOR DEUTSCHLAND UND ÖSTERREICH**
ComputerWorks GmbH
Schwarzwaldstraße 67
79539 Lörrach

Tel.: 07621 / 40 18 0
Fax: 07621 / 40 18 18

info@computerworks.de
www.computerworks.de

**Support:**

Für Vectorworks Service Select-Kunden:
www.vectorworks.de/serviceselect

Online-Supportformular:
www.computerworks.de/vwsupport

Telefon-Hotline:
Tel.: 09001 / 23 45 77 (Đ 1,95 pro Minute)

**DISTRIBUTOR SCHWEIZ**
ComputerWorks AG
Florenz-Strasse 1e
4142 Münchenstein

Tel.: 061 / 337 30 00
Fax: 061 / 337 30 01

info@computerworks.ch
www.computerworks.ch

**Support:**

Für Vectorworks Service Select-Kunden:
www.vectorworks.ch/serviceselect

Online-Supportformular:
www.computerworks.ch/vwsupport

Telefon-Hotline:
Tel.: 0900 337 337 (Fr. 3.– pro Minute für Anrufe ab Festnetz)

Vectorworks wird in Deutschland, Österreich und der Schweiz von ComputerWorks betreut.

ComputerWorks

AUTHORIZED DISTRIBUTOR

**HERSTELLER**
Vectorworks, Inc
7150 Riverwood Drive
Columbia, MD 21046-1295
USA

Tel.: 001 410 / 290 51 14
Fax: 001 410 / 290 8050

www.vectorworks.net

# INHALTSVERZEICHNIS

# EINLEITUNG

Mit dem Werkzeug **Marionette** (Werkzeuggruppe „Konstruktion“) steht Ihnen eine Benutzeroberfläche für Visual Scripting zur Verfügung. Diese ermöglicht es unerfahrenen Anwendern in Vectorworks komplexe Scripts zu erzeugen, ohne die zugrunde liegende Programmiersprache „Python“ verstehen zu müssen. Zu diesem Zweck werden sogenannte „Nodes“ (engl. für Knotenpunkte) platziert, die für verschiedene Aktionen stehen, die in Vectorworks stattfinden. Die Nodes werden in Aktionsnetzwerken angeordnet, die der Reihe nach ablaufen. Fortgeschrittenere Node-Typen ermöglichen es Marionette-Netzwerke wiederzuverwenden, zusammenzuführen oder weiterzugeben. Erfahrene Anwender können, wenn nötig, das zugrunde liegende Script bearbeiten. Es kann zwar nützlich sein, vor dem Gebrauch von Marionette das Programmieren mit Scripts zu verstehen, notwendig ist es jedoch nicht.

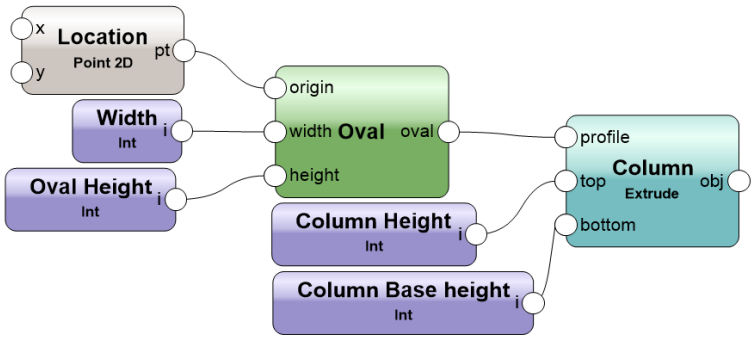


BILD 1 – Marionette-Script

Im Folgenden lernen Sie anhand von zwei Beispielen, wie mit Sie mit Marionette Scripts erzeugen. Im ersten Beispiel wird ein Befehl angelegt, mit dem Sie Klassennamen mehrerer Klassen gleichzeitig ändern können, im zweiten Beispiel wird ein Objekt (eine Rampe) erzeugt.

Weitere Informationen sowie Einführungsfilme und Übungsdateien für Marionette finden Sie unter folgendem Link: https://www.computerworks.de/produkte/vectorworks/vectorworks-architektur/marionette.html .

Ein (englischsprachiges) Forum, in dem Sie sich mit anderen Anwendern über das Schreiben von Scripts mit Marionette austauschen können finden Sie hier: https://forum.vectorworks.net/index.php?/forum/48-marionette/

Eine (englische) Galerie mit zahlreichen Marionette-Scripts von Vectorworks-Anwendern finden Sie hier: https://forum.vectorworks.net/index.php?/files/

# MARIONETTE-NODES EINFÜGEN

Wollen Sie das Marionette-Werkzeug in Vectorworks verwenden, müssen Sie kein Plug-in laden oder in einer separaten Benutzeroberfläche arbeiten. Das Marionette-Werkzeug sieht so aus  und befindet sich in der Werkzeugpalette „Konstruktion“. Klicken Sie einfach auf das Werkzeug und dann auf das Einblendmenü in der Methodenzeile. Verwenden Sie das Marionette-Werkzeug zum ersten Mal nach dem Start von Vectorworks, zeigt das Einblendmenü den Text „Wählen Sie eine Node Definition“ an. Nachdem Sie den ersten Node gewählt haben, wird standardmäßig der zuletzt verwendete Node angezeigt. Klicken Sie auf das Einblendmenü, öffnet sich das Zubehör-Auswahlmenü. Dort wird automatisch die Bibliotheksdatei angezeigt, die die Marionette-Nodes enthält.

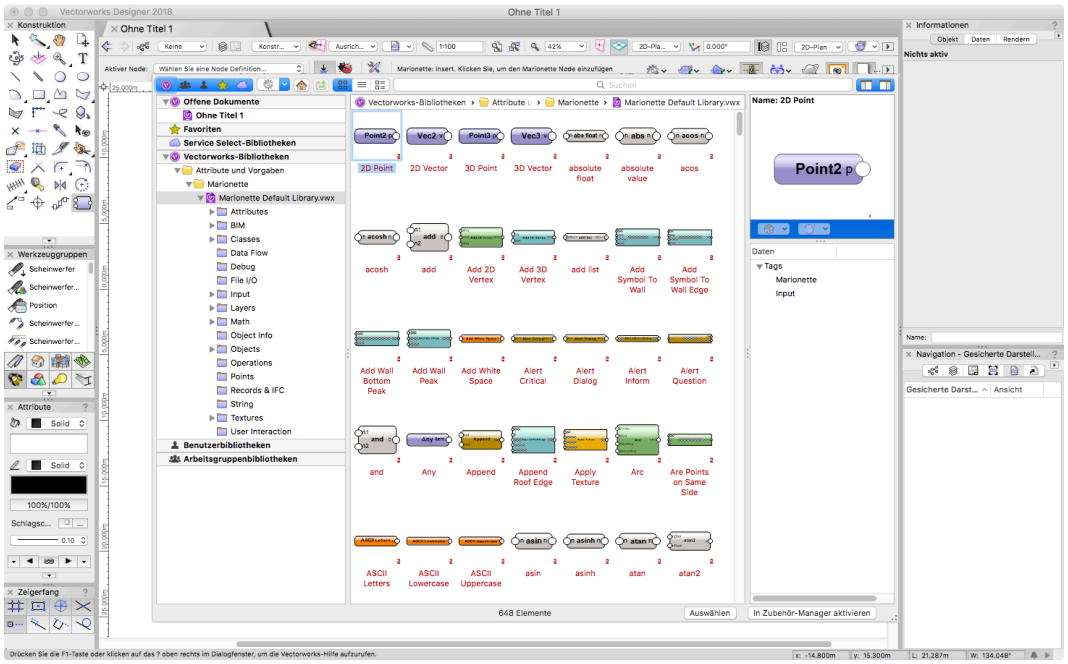


BILD 2 – Das Werkzeug Marionette mit geöffnetem Zubehör-Auswahlmenü

In den Ordnern der Marionette Vorgabe-Bibliothek („Marionette Default Library“) finden Sie alle verfügbaren Nodes, aufgeteilt in Kategorien. Sie können einen Node wählen, indem Sie durch die Ordner navigieren oder im Suchfeld des Zubehör-Auswahlmenüs seinen Namen eingeben. Doppelklicken Sie dann auf den Node und klicken Sie in die Zeichnung, um ihn einzufügen.

# MARIONETTE-NODES EINFÜGEN [FORTS.]

## NODES ZU NETZWERKEN VERBINDEN

Nodes werden zu Netzwerken miteinander verbunden, die ein funktionsfähiges Script erzeugen und die Befehle der einzelnen Nodes ausführen. Ein vollständiges Netzwerk ist ein Marionette-Script. Alle Scripts werden von links nach rechts gelesen und die Daten fließen in eine Richtung.

HINWEIS: Netzwerke können nur in der Ansicht „2D-Plan" erzeugt und bearbeitet werden.

Nodes werden folgendermaßen miteinander verbunden:

1. Aktivieren Sie das Aktivieren-Werkzeug und klicken Sie auf den Kontrollpunkt eines Node Output-Ports. Klicken Sie dann auf den Input-Port eines anderen Nodes.
2. Die beiden Nodes werden verbunden. Sie können die Outputs jedes Nodes auch mit mehreren Inputs anderer Nodes verbinden. Umgekehrt können mehrere Outputs auch mit einem einzelnen Input verbunden werden.

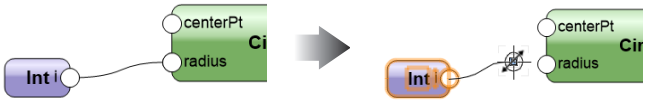


BILD 3 – Nodes verbinden

# EIGENE MARIONETTE-ZUBEHÖRBIBLIOTHEK HINZUFÜGEN

Das Marionette-Objekt, das in Tutorial 2 erzeugt wird, verwendet einen bereits erzeugten Wrapper Node. Ein Wrapper Node ist ein Node-Netzwerk, das in einem Paket zusammengepackt (wrapped = eingewickelt) ist und wie jeder andere Node als Teil eines größeres Netzwerks verwendet werden kann.

Sie können die Datei „Mein Marionette-Zubehoer“, die diesen Wrapper Node enthält, unter folgendem Link herunterladen:

http://cw-downloads.eu/white_paper/mein_marionette_zubehoer.vwx

Sichern Sie die Datei auf Ihrer Festplatte und öffnen Sie Vectorworks. Wählen Sie im Zubehör-Manager im Einblendmenü **Aktion** den Befehl **Neue Dokumente werden Favoriten**. Navigieren Sie zur heruntergeladenen Datei und klicken Sie auf **Öffnen**. Die Datei sollte jetzt im Navigationsbereich links im Zubehör-Manager angezeigt werden. Klicken Sie auf den Pfeil neben dem Dateinamen und aktivieren Sie den Ordner „Meine Wrapper“. In der Zubehörliste wird ein Wrapper Node angezeigt. Dieser kann jetzt in die Zeichnung eingesetzt werden.

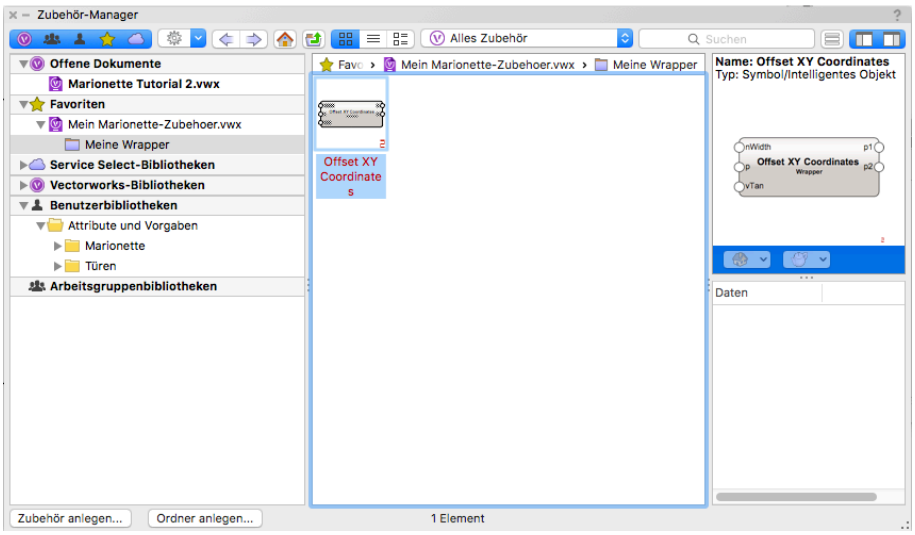


BILD 4 – Der Zubehör-Manager mit der Favoritendatei „Mein Marionette-Zubehoer“

# TUTORIAL 1: EINEN BEFEHL ERZEUGEN

Für dieses Tutorial erzeugen wir einen Menü-Befehl, der es Ihnen erlaubt Klassennamen für mehrere Klassen gleichzeitig zu ändern, basierend auf dem Text des Klassennamens – im Grunde ein Suchen-und-Ersetzen-Befehl für Klassen.

Da wir uns mit Klassen beschäftigen, fangen wir damit an, dass wir Nodes aus dem Ordner „Classes" holen. Aktivieren Sie dazu zuerst das Werkzeug **Marionette** in der Werkzeugpalette „Konstruktion" und stellen Sie sicher, dass die Methode „Einfügen" in der Methodenzeile aktiviert ist. Klicken Sie dann auf das Einblendmenü in der Methodenzeile. In der Marionette Default Library finden Sie einen Ordner namens „Classes". Dieser enthält alles Nodes, die für das Beeinflussen von Klassen in Vectorworks wichtig sind. Wir benötigen in unserem Netzwerk eine Liste der Klassen, suchen Sie also nach dem Node **Get Class List** im Ordner „Classes" → „Get Operations - Classes" und platzieren Sie diesen in der Zeichnung.


BILD 5 – Der Node **Get Class List** aus dem Ordner „Classes"

Wenn Sie den Node anschauen, können Sie sehen, dass der Input-Name „iIndex" und der Output-Name „sName" ist. Dies bedeutet, dass der Node einen Integer (ganze Zahl) benötigt, die eine Index-Zahl repräsentiert, und dass er einen String (Zeichenfolge) produziert, die einen Namen repräsentiert. Aktivieren Sie den Node und klicken Sie in der Infopalette auf **Beschreibung**. In der Beschreibung steht: „Returns the name of a class at a given index in the document class list." (Gibt den Namen einer Klasse wieder, wenn der vorgegebene Index in der Klassenliste des Dokuments vorkommt.) Um den Node verwenden zu können, benötigen wir eine Liste der Index-Zahlen, die jede Klasse im Dokument repräsentieren. Das bedeutet, dass wir einen Node brauchen, der uns sagt, wie viele Klassen im Dokument sind. Gehen Sie also wieder zum Werkzeug **Marionette** und suchen Sie nach dem Node **Get Class Count** im Ordner „Classes" → „Get Operations - Classes".


BILD 6 – Der Node **Get Class Count** aus dem Ordner „Classes"

Sie können die Zahl, die von diesem Node produziert wird, verwenden, um eine Reihe von Zahlen zu erzeugen.

Als Nächstes benötigen wir einen **Series**-Node aus dem Ordner „Data Flow".

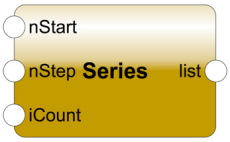


BILD 7 – Der Node **Series** aus dem Ordner „Data Flow"

# TUTORIAL 1: **EINEN BEFEHL ERZEUGEN** [FORTS.]

Wenn Sie die Beschreibung des **Series**-Nodes betrachten, steht dort, dass der vorgegebene Startwert „0“ ist, die vorgegebene Schrittgröße „1“ und der vorgegebene Zählwert „1“. In der Beschreibung des Nodes **Get Class List** steht, dass der „iIndex“-Input Zahlen von „1“ bis „n“ aufnimmt. Unsere Reihe muss deshalb mit einer „1“ beginnen. Aktivieren Sie also das Werkzeug **Marionette** und holen Sie einen **Integer**-Node (für ganze Zahlen) aus dem Ordner „Input“ → „Basic“


BILD 8 – Der Node **Integer** aus dem Ordner „Basic“

Dieser Node produziert einen Ganzzahl-Wert, den der Anwender in das entsprechende Feld in der Infopalette eingibt. Aktivieren Sie also den Node und tippen Sie in der Infopalette im Feld **Integer** den Wert „1“ ein. Verbinden Sie dann den **Integer**-Node mit dem „nStart“-Input des **Series**-Nodes und verbinden Sie den Node **Get Class Count** mit dem „iCount“-Input des **Series**-Nodes. Danach verbinden Sie den Output des **Series**-Nodes mit dem Nodes **Get Class List**.

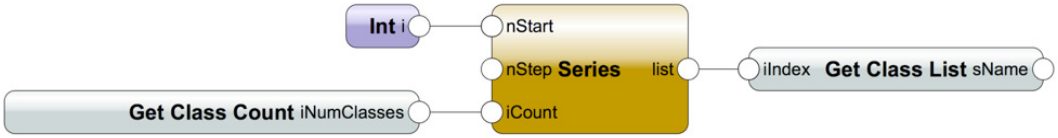


BILD 9 – Eine Liste mit Klassennamen erzeugen

Diese vier Nodes erzeuge eine Liste aller Klassennamen im aktuellen Dokument. Wenn Sie sehen wollen, ob Ihr Netzwerk richtig funktioniert, aktivieren Sie das Werkzeug **Marionette**, nehmen den Node **Dummy End** aus dem Ordner „Debug“ und verbinden den Output des Nodes **Get Class List** damit.


BILD 10 – Der Node **Dummy End** aus dem Ordner „Debug“

# TUTORIAL 1: **EINEN BEFEHL ERZEUGEN** [FORTS.]

Der Zweck des Nodes **Dummy End** ist es, einfach eine Verbindung zu erzeugen, die für die Fehlerbehebung verwendet werden kann. Aktivieren Sie das Werkzeug **Marionette** sowie dessen Methode „Debug-Modus“  in der Methodenzeile. Bewegen Sie Ihren Zeiger zurück in den Zeichnungsbereich, sollte ein Handzeiger angezeigt werden. Bewegen Sie diesen über die Verbidnung, die den Node **Get Class List** mit dem Node **Dummy End** verbindet. Wird die Verbindung rot markiert, klicken Sie darauf, um das Netzwerk auszuführen. Das Dialogfenster „Marionette Wire Values“ öffnet sich und zeigt die Werte, die durch die Verbindung laufen. In diesem Fall ist dies eine List der aktuellen Klassen im Dokument in der Reihenfolge, in der sie erzeugt wurden. Die Zahlen in den Klammern auf der linken Seite zeigen die Position jedes Werts in der Liste an, mit anderen Worten, einen Index jedes Werts. Sind mehr als 10 Werte in der Liste, können Sie mit den Knöpfen „Vorwärts“ und „Zurück“ durch die Liste blättern, immer 10 Werte auf einmal. Wenn Sie ein Netzwerk im Debug-Modus ausführen, erscheint eine Zahl auf der rechten Seite jedes Output-Ports, um zu zeigen, wie viele Werte aus diesem Port kommen.

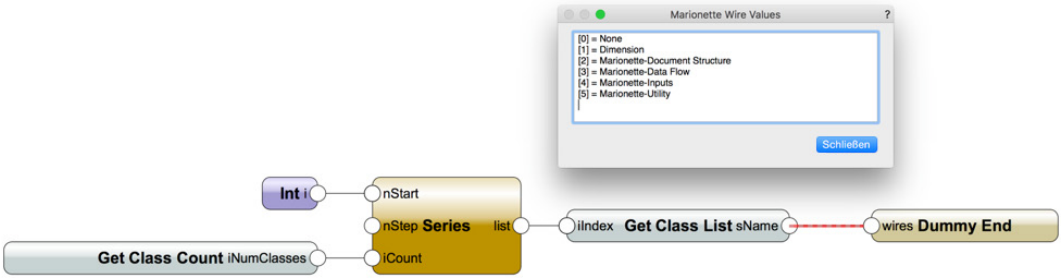


BILD 11 – Anzeige der Werte, die durch eine Verbindung laufen, im Debug-Modus

Lösen Sie die Verbindung des Nodes **Dummy End** wieder und schieben Sie für den späteren Gebrauch auf die Seite oder löschen Sie ihn.

Als Nächstes benötigen wir einen Weg unsere Klassenliste nach genau den Klassen zu filtern, die wir bearbeiten wollen. Deshalb benötigen wir einen **Filter**-Node und einen Node für eine „Wenn-Aussage“. Den **Filter**-Node finden Sie im Ordner „Data Flow“. Platzieren Sie ihn in der Zeichnung.


BILD 12 – Der Node **Filter** aus dem Ordner „Data Flow“

# TUTORIAL 1: **EINEN BEFEHL ERZEUGEN** [FORTS.]

Jetzt benötigen wir einen Node **If In String** aus dem Ordner „String“. Der Node ist **If In String** ist ein Bedingungsnode, der testet, ob ein String in einem anderen existiert.

**Hinweis:** Wird dieser Node nicht im Zubehör-Auswahlmenü angezeigt, müssen Sie Ihre Vectorworks-Bibliotheken aktualisieren.


BILD 13 – Der Node **If in String** aus dem Ordner „String“

Verbinden Sie als Nächstes den Output von **Get Class List** mit dem Input „sFull“ des Nodes **If In String** sowie mit dem Input „item“ des **Filter**-Nodes und verbinden Sie den Output „b“ des Nodes **If In String** mit dem Input „b“ des **Filter**-Nodes.

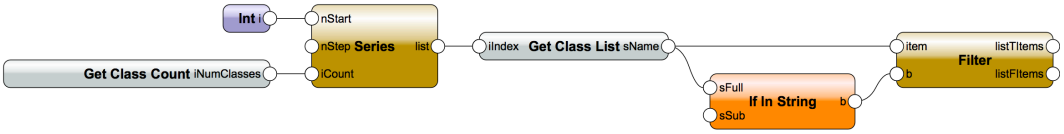


BILD 14 – Der aktuelle Stand des Netzwerks

Jetzt benötigen wir einen String, um innerhalb der Klassennamen zu suchen. Dazu brauchen wir einen **String**-Node aus dem Ordner „Input“ → „Basic“ und einen Node **String Dialog** aus dem Ordner „User Interaction“.


BILD 15 – Der **String**-Node aus dem Ordner „Input“ und der Node **String Dialog** aus dem Ordner „User Interaction“

# TUTORIAL 1: **EINEN BEFEHL ERZEUGEN** [FORTS.]

Platzieren Sie diese Nodes in der Zeichnung und verbinden Sie den Output des **String**-Nodes mit den Input des Nodes **String Dialog** und dann den Output des Nodes **String Dialog** mit dem Input „sSub“ des Nodes **If In String**. Aktivieren Sie den Node **String Dialog**. Löschen Sie in der Infopalette unter **Request** den Text „Enter a string“ und geben Sie stattdessen „Wenn der Klassenname den folgenden Text enthält“ ein. Für den **String**-Node können Sie das Textfeld leer lassen. Der Node **String Dialog** erlaubt es Stringwerte einzutippen, nachdem das Netzwerk ausgeführt wurde. Diese Art Node ist ideal für Befehle. Führen Sie das Netzwerk aus, um den Node **String Dialog** in Aktion zu sehen.

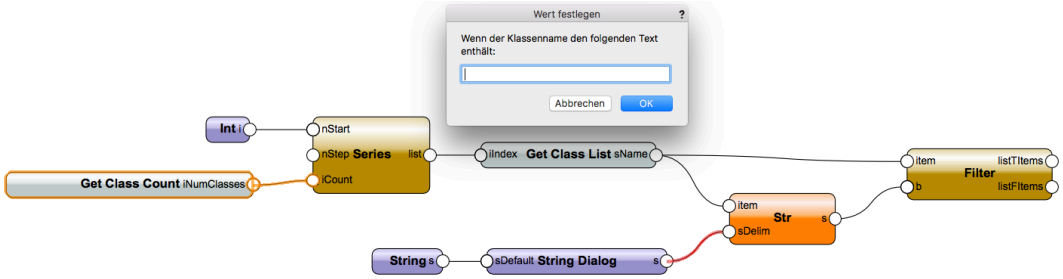


BILD 16 – Führen Sie das Netzwerk an diesem Punkt aus, erscheint ein Dialogfenster, in das ein String-Wert eingegeben werden muss.

Wenn Sie dieses Netzwerk ausführen, öffnet sich ein Dialogfenster, in dem Sie einen String-Wert (Text) eingeben können, nach dem in den Klassennamen-Strings gesucht wird.

Aufgrund des bisher erzeugten Netzwerks besteht die Ausgabe des **Filter**-Nodes aus zwei Listen, einer Liste mit Klassen, die den im Dialogfenster eingegebenen String enthalten, und eine Liste mit Klassen, die den String nicht enthalten. Die Liste mit Klassen, die den String enthalten, wird mit den Boolschen „Wahr“-Werten aus dem Node **If In String** repräsentiert, deshalb kommen Sie aus dem Output „listTItems“ (T für „True“). Wählen Sie jetzt im Ordner „String“ den Node **Replace String**.


BILD 17 – Der Node **Replace String** aus dem Ordner „String“

Haben Sie diesen Node in der Zeichnung platziert, gehen Sie zum Ordner „Classes“ → „Set Operations - Classes“, wählen Sie den Node **Rename Class** und platzieren Sie diesen ebenfalls in der Zeichnung.


BILD 18 – Der Node **Rename Class** aus dem Ordner „Classes“

# TUTORIAL 1: **EINEN BEFEHL ERZEUGEN** [FORTS.]

Verbinden Sie den Output „listTItems“ des **Filter**-Nodes mit dem Input „sFull“ des Nodes **Replace String** sowie mit dem Input „sName“ des Nodes **Rename Class**. Als Nächstes verbinden Sie den Output des Nodes **Replace String** mit dem Input „sNewName“ des Nodes **Rename Class**. Jetzt brauchen wir nur noch die Input-Werte für die anderen Inputs des Nodes **Replace String**. Der Output des Nodes **String Dialog** kann jetz mit dem Input „sOld“ des Nodes **Replace String** verbunden werden, da der String, nach dem Sie in der Klasse suchen, derselbe String ist, den Sie ersetzen.

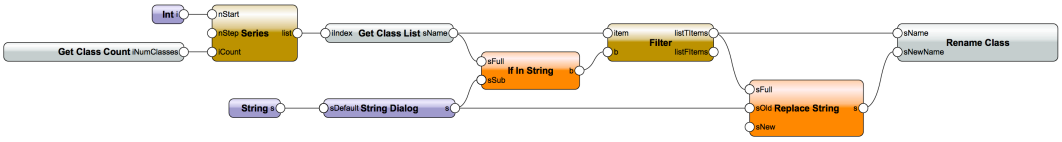


BILD 19 – Das Netzwerk mit dem Node **Rename String** und dem Node **Rename Class**

Als Nächstes können wir die Nodes **String** und **String Dialog** kopieren und direkt unter den Originalen einfügen. Dann verbinden wir den Output des neuen Nodes **String Dialog** mit dem Input „sNew“ des Nodes **Replace String**. Aktivieren Sie den neuen Node **String Dialog** und ändern Sie den Text im Textfeld der Infopalette zu „Mit diesem Text ersetzen:“, denn hier geben Sie den Ersatztext für die Klassennamen ein. Ihr Netzwerk sollte folgendermaßen aussehen:

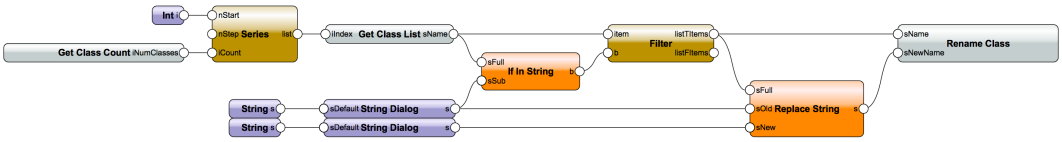


BILD 20 – Das Netzwerk kann jetzt in einen Befehl konvertiert werden.

Testen Sie das Netzwerk, um sicherzustellen, dass es funktioniert. Wenn Sie das Netzwerk ausführen, sollten Sie dazu aufgefordert werden den Text einzugeben, nach dem gesucht wird, und dann den neuen Text einzugeben, der den alten Text ersetzt.

Jetzt können wir unser Netzwerk in einen Befehl konvertieren. Zuerst müssen wir das Netzwerk wrappen, indem wir mit der rechten Maustaste auf einen beliebigen Node klicken und im Kontextmenü **Wrap Marionette Network** wählen. Wir werden dazu aufgefordert, einen Namen und eine Beschreibung für unseren Wrapper einzugeben. Sie können ihn „Klassen umbenennen“ nennen und eine kurze Beschreibung seiner Funktion eingeben.

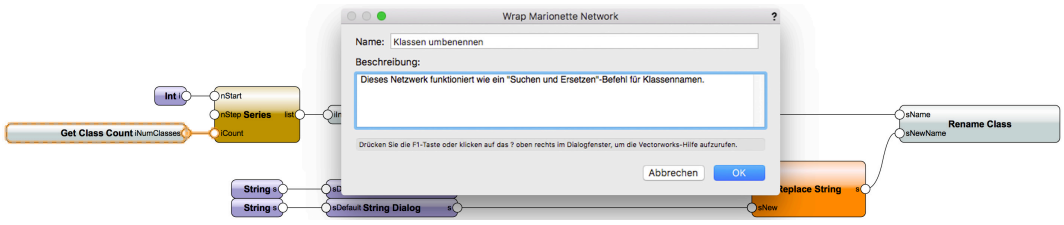


BILD 21 – Das Dialogfenster „Wrap Marionette Network“

# TUTORIAL 1: **EINEN BEFEHL ERZEUGEN** [FORTS.]

Auch wenn die Wrapper-Informationen nicht mehr verfügbar sind, wenn der Wrapper in einen Befehl umgewandelt wurde, existiert der Wrapper in Ihrem Benutzerordner und wenn Sie ihn mit jemandem teilen wollen, sind der Name und die Beschreibung nützlich. Klicken Sie mit der rechten Maustaste auf den Wrapper und wählen Sie im Kontextmenü **In Menübefehl umwandeln**. Geben Sie den Befehlsnamen im erscheinenden Dialogfenster ein. Verwenden Sie genau den Namen, der im Menü erscheinen soll. In Vectorworks werden Befehle, bei denen Dialogfenster erscheinen, mit drei Punkten nach dem Befehlsnamen geschrieben. Geben Sie also „Klassen umbenennen…“ ein und klicken Sie auf **OK**. Wählen Sie jetzt **Extras → Marionette** und sehen Sie dort Ihren neu erzeugten Befehl.

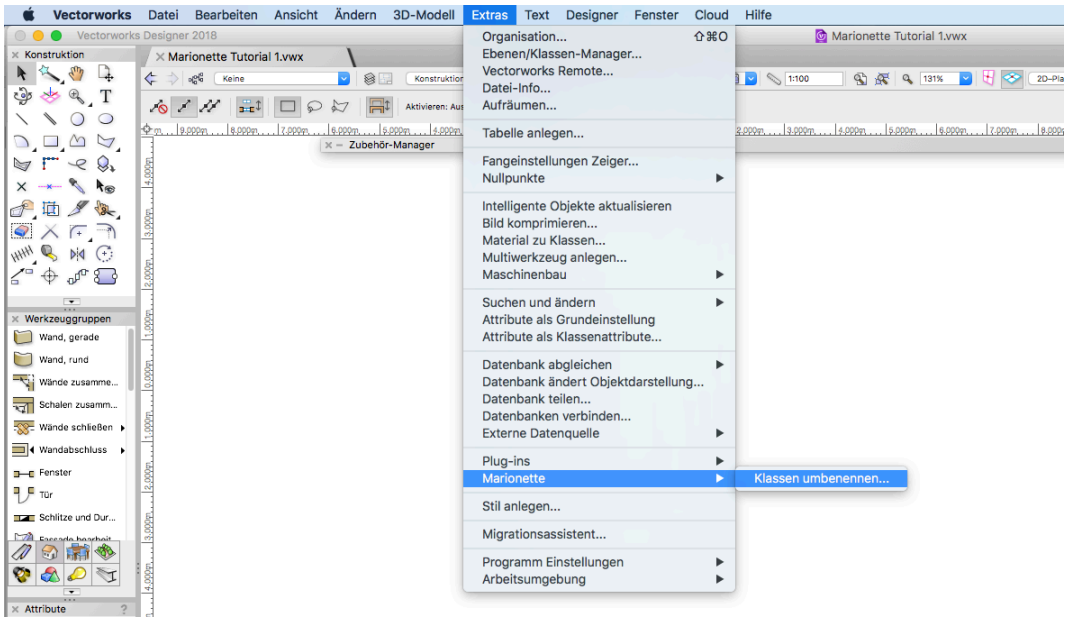


BILD 22 – Der Pfad zum neuen Befehl

# TUTORIAL 1: **EINEN BEFEHL ERZEUGEN** [FORTS.]

Wollen Sie den Befehl bearbeiten oder mit anderen teilen, können Sie ihn in Ihrem Vectorworks-Benutzerordner unter Bibliotheken → Attribute und Vorgaben → Marionette → Marionette Command Library.vwx finden. Wollen Sie einen Befehl bearbeiten, bearbeiten Sie den Wrapper wie jeden anderen Wrapper und wandelnt Sie ihn dann wieder in einen Befehl um, geben Sie diesem denselben Namen wie dem alten Befehl und ersetzen Sie den alten durch den neuen Befehl.

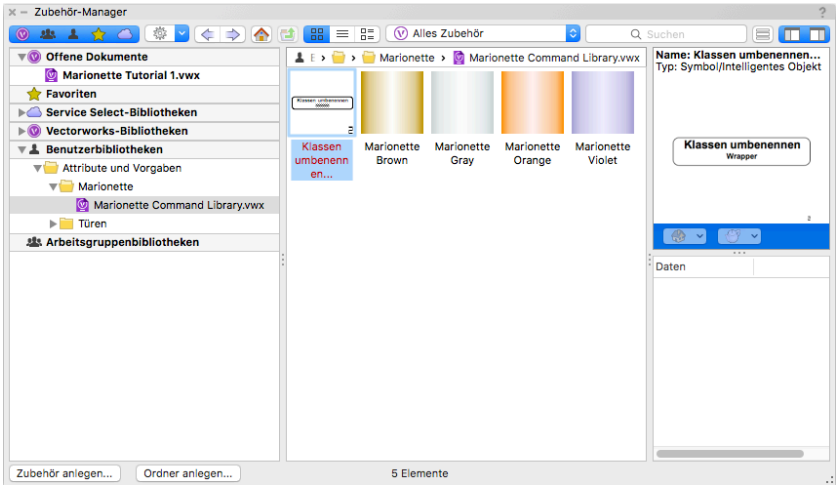


BILD 23 – Der Speicherort des Wrappers nach der Konvertierung in einen Befehl

# TUTORIAL 2: **EIN MARIONETTE-OBJEKT ERZEUGEN**

Für dieses Tutorial werden wir ein Marionette-Objekt erzeugen, das als intelligente, pfad-basierte Rampe dient. Wir beginnen mit einem gezeichneten Stück Geometrie. Zeichnen Sie eine gebogene Polylinie, die ca. 15 m lang ist. Sie können den Maßstab Ihrer Zeichnung auf 1:50 setzen.

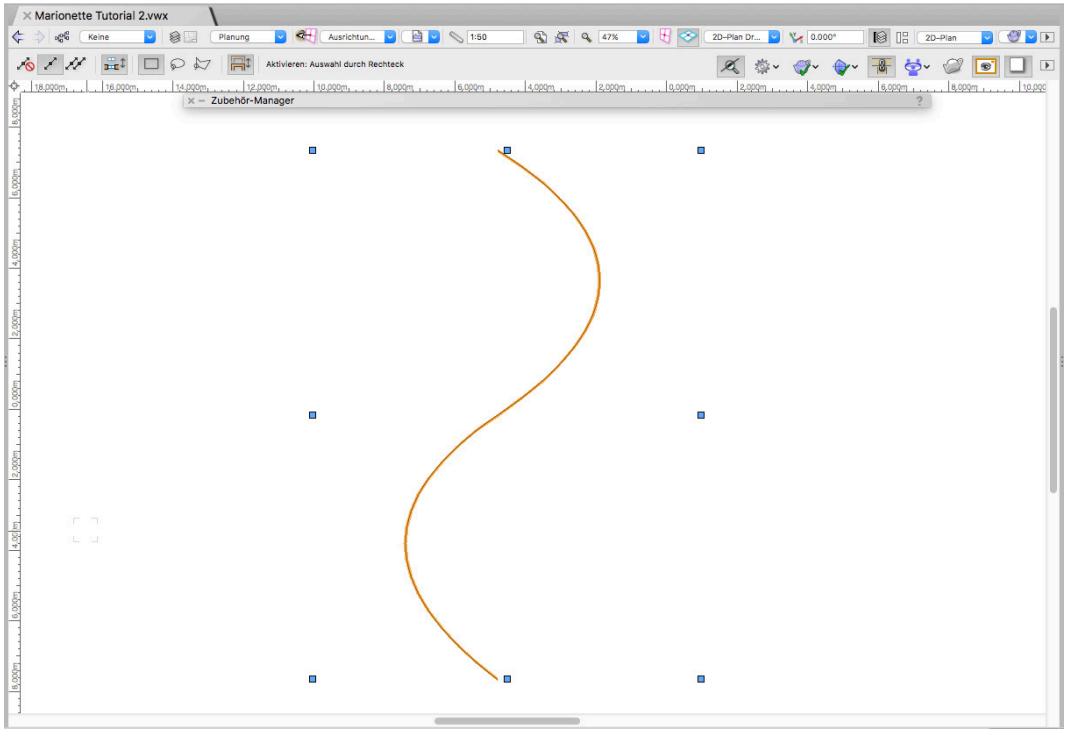


BILD 24 – Die Startgeometrie für das Marionette-Objekt

Marionette-Nodes werden automatisch an den Maßstab angepasst, so dass es einfach ist den Pfad und die Marionette-Nodes gleichzeitig anzuzeigen, so lange der Maßstab Ihres Dokuments für die Geometrie richtig ist. Damit Marionette auf eine bestehende Geometrie Bezug nehmen kann, muss diese einen Namen erhalten. Aktivieren Sie dazu das Objekt und geben Sie unten in der Infopalette im Feld „Name“ einen einzigartigen Namen ein. Aktivieren Sie dann das Werkzeug **Marionette** und wählen Sie den Node **Name** aus dem Ordner „Input“ → „Basic“ und setzen Sie diesen in die Zeichnung ein.


BILD 25 – Der **Name**-Node aus dem Ordner „Input“

# TUTORIAL 2: **MARIONETTE-OBJEKT ERZEUGEN** [FORTS.]

In der Infopalette des **Name**-Nodes finden Sie ein Feld namens **Object Name**. Geben Sie dort den einzigartigen Namen ein, den Sie der Polylinie gegeben haben, und schalten Sie die Option **Create a duplicate object** aus. Diese Option sollte nur aktiviert sein, wenn Sie planen das Objekt innerhalb eines Netzwerks zu löschen.

Jetzt haben wir Geometrie mit einem Marionette-Netzwerk verbunden und können sie in einem Netzwerk verwenden. Der **Name**-Node ist notwendig, um die Polylinie mit dem Netzwerk zu verbinden, während wir dieses aufbauen und testen, aber wenn wir das Netzwerk in ein Marionette-Objekt umwandeln, müssen wir den **Name**-Node durch einen Node **Control Geometry** ersetzen. Um diesen Transfer zu vereinfachen, können wir den **Name**-Node mit einem **Pass**-Node aus dem Ordner „Data Flow“ verbinden.


BILD 26 – Der **Name**-Node aus dem Ordner „Input“ wird verbunden mit einem **Pass**-Node aus dem Ordner „Data Flow“.

Da die Polylinie als Grundlage für das Netzwerk dient, wird der **Name**-Node wahrscheinlich mit vielen anderen Nodes verbunden, und wenn er dann ersetzt werden muss, müssen alle diese Verbindungen auch ersetzt werden. Der **Pass**-Node tut nichts, er erlaubt einfach, dass die Informationen aus den Verbindungen durch ihn hindurchlaufen. Wenn wir den Output-Port des **Pass**-Nodes dazu verwenden, den **Name**-Node mit dem Rest des Netzwerks zu verbinden, müssen wir beim Ersetzen des **Name**-Nodes nur den **Name**-Node vom **Pass**-Node lösen und den Node **Control Geometry** mit dem **Pass**-Node verbinden.

Die Polylinie dient als Pfad oder Mittellinie unserer Rampe. Um die Kurve allmählich anzuheben, müssen wir zuerst einen Bereich von Koordinaten entlang des Pfads festlegen. Dazu benötigen wir zuerst einen Node **Get Point On Poly** aus dem Ordner „Objects“ → „Poly“ → „2D“.


BILD 27 – Der Node **Get Point On Poly** aus dem Ordner „2D“

Wie Sie in der Beschreibung des Nodes sehen können, gibt dieser einen Punkt auf einer Polylinie aus, basierend auf einer bestimmten Distanz, sowie eine Vektortangente zu diesem Punkt. Um eine Reihe von Punkten entlang der Polylinie zu erhalten, die gleich weit voneinander entfernt sind, müssen wir einen Bereich von Werten zwischen 0 und der Gesamtlänge der Polylinie erzeugen. Dazu benötigen wir einen Node **Get Length** aus dem Ordner „Object Info“ und einen **Range**-Node aus dem Ordner „Data Flow“.

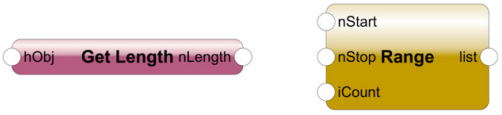


BILD 28 – Der Node **Get Lenght** aus dem Ordner „Objekt Info“ und der **Range**-Node aus dem Ordner „Data Flow“

# TUTORIAL 2: **MARIONETTE-OBJEKT ERZEUGEN** [FORTS.]

Als Nächstes benötigen wir einen **Integer**-Node aus dem Ordner „Input“ → „Basic“.


BILD 29 – Der **Integer**-Node aus dem Ordner „Input“

Aktivieren Sie den **Integer**-Node, tippen Sie „20“ im Datenfeld der Infopalette ein und verbinden Sie den Output des **Integer**-Nodes mit dem Input „iCount“ des **Range**-Nodes. Die Zahl 20 ist ein willkürlicher Wert, mit dem die Anzahl der Punkte entlang der Polylinie erzeugt wird. Beachten Sie, dass der Input „iCount“ für den **Range**-Node eigentlich die Anzahl der Teilungen zwischen Werten ist, nicht die Anzahl der Werte, so dass ein Bereich mit der Zahl 20 als Resultat 21 Werte hat. Verbinden Sie als Nächstes den Output des **Pass**-Nodes mit dem Input des Nodes **Get Length** und dann den Output des Nodes **Get Length** mit dem Input „nStop“ des **Range**-Nodes. Verbinden Sie dann den Output des **Range**-Nodes mit dem Input „nDist“ des Nodes **Get Point On Poly**. Verbinden Sie schließlich den Output des **Pass**-Nodes mit dem Input „hPoly“ des Nodes **Get Point On Poly**.

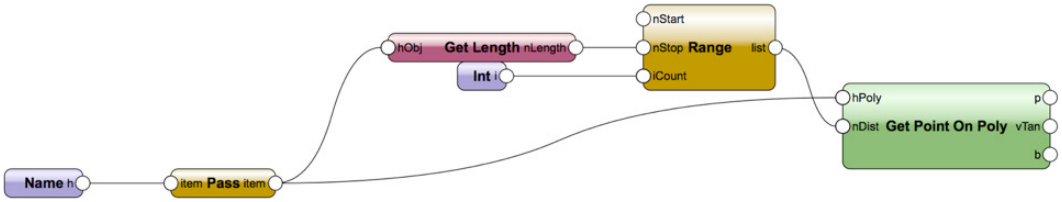


BILD 30 – Einen Bereich von Werten erzeugen, um Punkte entlang einer Polylinie zu definieren

Wenn ein Bereich erzeugt wird, der auf der Länge der Polylinie basiert, wird immer ein „Falsch“-Wert für den letzten Punkt erzeugt, wahrscheinlich wegen der Abweichungen des Nodes **Get Point On Poly**. Die Lösung dafür ist, den letzten Wert der Liste zu entfernen und durch den Endpunkt der Polylinie zu ersetzen, den wir einfach finden können. Zu diesem Zweck brauchen wir einen Node **Get 2D Vertex** aus dem Ordner „Objects“ → „Poly“ → „2D“.


BILD 31 – Der Node **Get 2D Vertex** aus dem Ordner „2D“

Dieser Node gibt uns einen Scheitelpunkt, basierend auf dessen Indexwert. Um den Indexwert des letzten Punkts zu erhalten, müssen wir wissen, wie viele Punkte in der Polylinie sind. Wir brauchen einen Node **Get 2D Vertex Count** aus dem Ordner „Objects“ → „Poly“ → „2D“ sowie einen Node **User Function** aus dem Ordner „Math“ → „Special“.


BILD 32 – Der Node **Get 2D Vertex Count** aus dem Ordner „2D“ und der Node **User Function** aus dem Ordner „Math“

# TUTORIAL 2: **MARIONETTE-OBJEKT ERZEUGEN** [FORTS.]

Der Node **Get 2D Vertex Count** gibt uns die Anzahl der Scheitelpunkte in der Polylinie, aber da Listen einen Startindex von 0 haben, ist der Indexwert des letzten Scheitelpunkts immer eins weniger als die Gesamtzahl der Scheitelpunkte. Hier kommt der Node **User Function** ins Spiel. Der Node **User Function** erlaubt es Ihnen eine Funktionen mit einer Variable in das Datenfeld der Infopalette einzugeben. Dabei kann es sich um eine Python-Funktion oder eine mathematische Funktion handeln, aber mathematische Funktionen müssen eine Python-Syntax verwenden. Dies ist einfach, auch wenn Sie Python nicht kennen, da dabei dieselben Konventionen gelten wie bei geschriebenen Funktionen. Die Funktion, die wir in die Infopalette des Nodes **User Function** eingeben ist „x-1“.

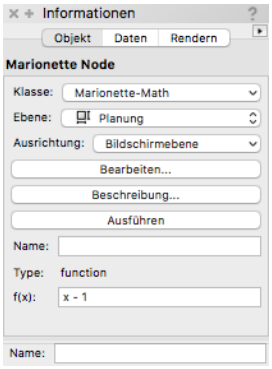


BILD 33 – Aktiveren Sie den Node **User Function** und tippen Sie „x-1“ in das Datenfeld „f(x)“ der Infopalette.

Verbinden Sie jetzt den **Pass**-Node mit dem Input des Nodes **Get 2D Vertex Count**, verbinden Sie den Output des Nodes **Get 2D Vertex Count** mit dem Input des Nodes **User Function** und verbinden Sie den Output des Nodes **User Function** mit dem Input „iIndex“ des Nodes **Get 2D Vertex**. Verbinden Sie dann den Output des **Pass**-Nodes mit dem Input „hPoly“ des Nodes **Get 2D Vertex**.

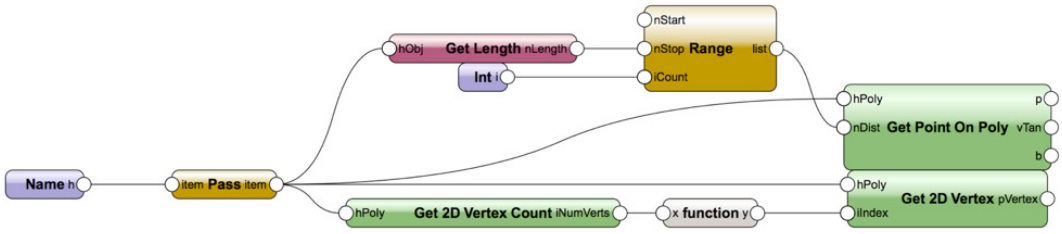


BILD 34 – Ein Netzwerk, das eine Reihe von Punkten entlang einer Polylinie sowie den letzten Scheitelpunkt des Polygons findet

# TUTORIAL 2: **MARIONETTE-OBJEKT ERZEUGEN** [FORTS.]

Jetzt haben wir eine Liste von Punkten entlang einer Polylinie sowie den letzten Scheitelpunkt der Polylinie und somit können den letzten Wert in der Punkteliste durch den letzten Scheitelpunkt ersetzen. Dafür benötigen wir den Node **Pop Back** und einen **Append**-Node. Beide Nodes finden Sie im Ordner „Data Flow“.


BILD 35 – Die Nodes **Pop Back** und **Append** aus dem Ordner „Data Flow“

Der Node **Pop Back** entfernt den letzten Wert aus der Input-Liste und der **Append**-Node fügt das Input-Element (item) am Ende der Input-Liste ein. Verbinden Sie den Output „p“ des Nodes **Get Point Only** mit dem Input „list“ des Nodes **Pop Back** und den Output „list“ des Nodes **Pop Back** mit dem Input „list“ des **Append**-Nodes. Verbinden Sie schließlich den Output des Nodes **Get 2D Vertex** mit dem Input „item“ des **Append**-Nodes.

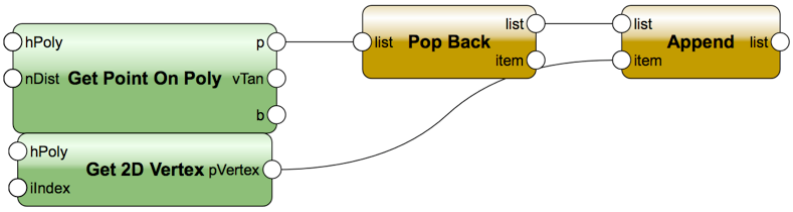


BILD 36 – Dieser Teil des Netzwerks entfernt den letzten Werte der Punkteliste und ersetzt ihn durch den letzten Punkt der Polylinie

Jetzt haben wir Punkt die gleich weit voneinander entfernt entlang der Polylinie verlaufen und müssen diese Punkte auf beiden Seiten der Polylinie versetzen, so dass unsere Rampe eine Breite erhält. Dazu benötigen wir den Wrapper **Offset XY Coordinates** aus der Datei „Mein Marionette-Zubehoer.vwx“, die sich im Zubehör-Manager bei den Favoriten befinden sollte (siehe Kapitel **Eigene Marionette-Zubehörbibliothek hinzufügen**). Klicken Sie im Einblendmenü **Aktiver Node** in der Methodenzeile im Navigationsbereich unter „Favoriten“ auf das Dreieck neben der Datei „Mein Marionette-Zubehoer.vwx“ und aktivieren Sie den Ordner „Meine Wrapper“. Wählen Sie dort den Wrapper **Offset XY Coordinates**.


BILD 37 – Der Wrapper **Offset XY Coordinates** aus dem Ordner „Meine Wrapper“ in der Favoritendatei „Mein Marionette-Zubehoer.vwx“

# TUTORIAL 2: **MARIONETTE-OBJEKT ERZEUGEN** [FORTS.]

Ein Wrapper ist ein Node-Netzwerk, das in einem Paket zusammengepackt (wrapped = eingewickelt) ist und wie jeder andere Node verwendet werden kann. Input- und Output-Ports werden erzeugt, indem Sie einfach Nodes mit offenen Ports einen Namen geben. Diese Ports erscheinen an der Außenseite des Wrappers. Der Wrapper **Offset XY Coordinates** besteht aus Math- und Point-Nodes, die die Abstände der x- und y-Werte eines Punkts berechnen, basierend auf einer Gesamtbreite und einem Tangentvektor. Doppelklicken Sie einfach auf den Wrapper, um das Netzwerk darin zu sehen. Mit Wrappern lassen sich Netzwerke einfach verdichten und für die Verwendung als Teil von anderen Netzwerken sichern.

Um den Wrapper **Offset XY Coordinates** in das Netzwerk einzubauen, werden der Output des **Append**-Nodes mit dem Input „p“ des Wrappers und der Output „vTan“ des Nodes **Get Point On Poly** mit dem Input „vTan“ des Wrappers verbunden. Die andere Information, die für den Wrapper **Offset XY Coordinates** benötigt wird, ist die Gesamtbreite der Rampe. Für diese Information benötigen wir einen **Dimension**-Node aus dem Ordner „Input“. Platzieren Sie den Node in der Zeichnung und geben Sie ihm den Wert „120 cm“ im Datenfeld **Dim** sowie den Namen „Rampenbreite“.


BILD 38 – Der **Dimension**-Node dem Ordner „Input“

Verbinden Sie den Output des **Dimension**-Nodes mit dem Input „nWidth“ des Wrappers **Offset XY Coordinates**.

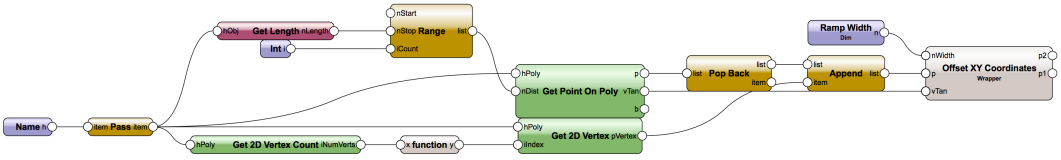


BILD 39 – Dieses Netzwerk erzeugt versetzte Punkte zur ursprünglichen Polylinie

# TUTORIAL 2: **MARIONETTE-OBJEKT ERZEUGEN** [FORTS.]

Jetzt haben wir unsere Punkte und können diese verwenden, um NURBS-Kurven zu bauen und diese Kurven dann hochzuziehen, um eine Rampe zu erzeugen. Zuerst müssen wir unsere beiden Gruppen von 2D-Punkten in vier Gruppen von 3D-PUnkten umwandeln (zwei Gruppen von 3D-Punkten, die der Steigung der Rampe folgen, und zwei Gruppen, die auf der z-Höhe 0 bleiben). Deshalb benötigen wir zwei Nodes **Get XY** und vier Nodes **Point 3D** aus dem Ordner „Points“. Verbinden Sie den Output „nX“ des ersten Nodes **Get XY** mit den Inputs „nX“ der ersten und zweiten Nodes **Point 3D** und den Output „nY“ des ersten Nodes **Get XY** mit den Inputs „nY“ der ersten und zweiten Nodes **Point 3D**. Wiederholen Sie das mit dem zweiten Node **Get XY** und den dritten und vierten Nodes **Point 3D**.

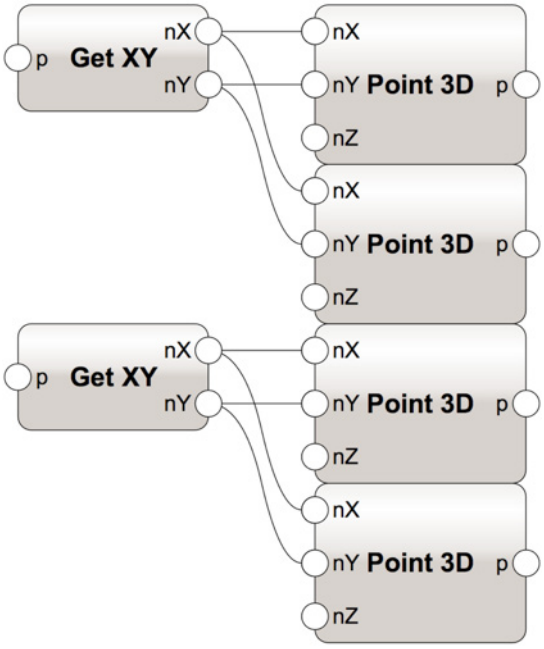


BILD 40 – Die Nodes **Get XY** verbunden mit den Nodes **Point 3D**, beide aus dem Ordner „Points“

Als Nächstes verbinden Sie den Output „p1“ des Wrappers **Offset XY Coordinates** mit dem Input des ersten Nodes **Get XY** und den Output „p2“ mit dem zweiten Node **Get XY**. Mit diesen Nodes können wir z-Werte zu unseren Punkten hinzufügen.

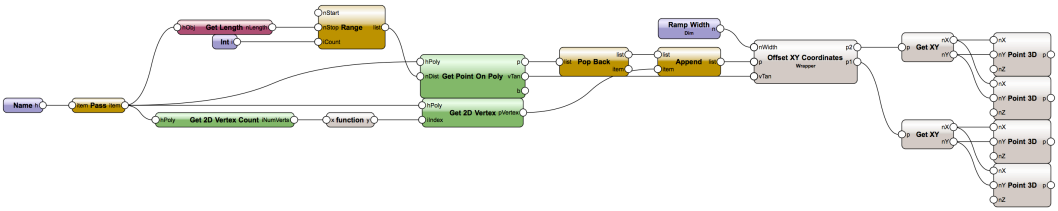

BILD 41 – Der aktuelle Stand des Netzwerks

# TUTORIAL 2: **MARIONETTE-OBJEKT ERZEUGEN** [FORTS.]

Um diese z-Werte zu erzeugen, benötigen wir zwei weitere **Dimension**-Nodes aus dem Ordner „Input“ → „Basic“, einen **Add**-Node aus dem Ordner „Math“ → „Basic“ sowie einen weiteren **Range**-Node aus dem Ordner „Data Flow“.

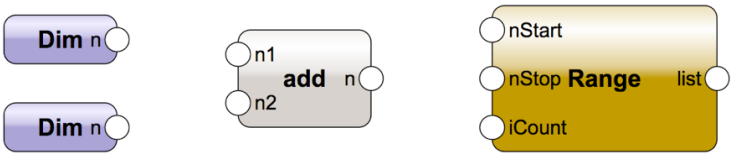


BILD 42 – Zwei **Dimension**-Nodes aus dem Ordner „Input“, der **Add**-Node aus dem Ordner „Math“ und der **Range**-Node aus dem Ordner „Data Flow“

Die **Dimension**-Nodes werden die Höhe repräsentieren, auf der die Rampe beginnt und die Höhe, bis zu der die Rampe steigt. Nennen Sie deshalb den ersten **Dimension**-Node „Starthöhe Rampe“ und geben Sie ihm den Wert 0 im Datenfeld **Dim**. Den zweiten **Dimension**-Node nenen Sie „Steigung Rampe“ und geben ihm den Wert 90 cm im Datenfeld **Dim**.

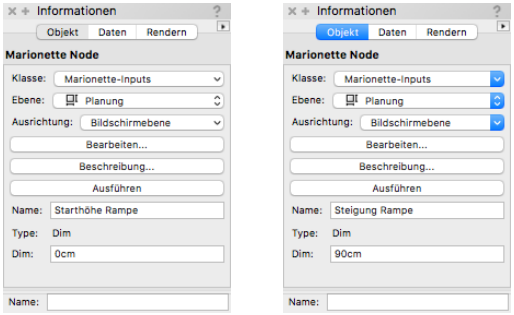


BILD 43 – Die Infopaletten der beiden **Dimension**-Nodes

Verbinden Sie „Starthöhe Rampe“ mit dem Input „nStart“ des **Range**-Nodes und mit dem Input „n1“ des **Add**-Nodes. Verbinden Sie dann „Steigung Rampe“ mit dem Input „n2“ des **Add**-Nodes und verbinden Sie den Output des **Add**-Nodes mit dem Input „nStop“ des **Range**-Nodes.

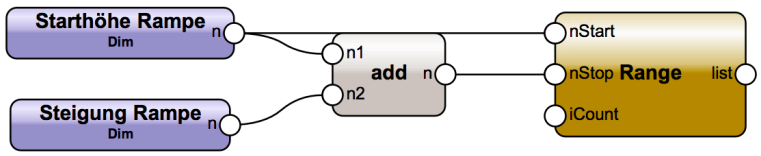


BILD 44 – Einen Bereich von z-Werten erzeugen

# TUTORIAL 2: **MARIONETTE-OBJEKT ERZEUGEN** [FORTS.]

Die „Starthöhe Rampe“ repräsentiert die Höhe, auf der das untere Ende der Rampe beginnt und ist deshalb der niedrigste Wert in unserem Bereich von z-Werten. Die „Steigung Rampe“ repräsentiert den Höhenunterschied zwischen dem Startpunkt und dem Endpunkt der Rampe. Deshalb ist „Starthöhe Rampe“ + „Steigung Rampe“ der höchste Wert in unserem Bereich von z-Werten. Jetzt müssen wir die festlegen, wie unser Bereich geteilt wird, und da wir diese Werte an die Anzahl der Teilungen unserer Polylinie anpassen, verwenden wir denselben „iCount“-Wert, den wir auf den ersten **Range**-Node in unserem Netzwerk angewendet haben. Verbinden Sie also den **Integer**-Node, der bereits im Netzwerk verwendet wird, mit dem Input „iCount“ des neuesten **Range**-Nodes. Verbinden Sie dann den Output dieses **Range**-Nodes mit dem Input „nZ“ des zweiten und dritten Nodes **Point 3D** im Netzwerk.

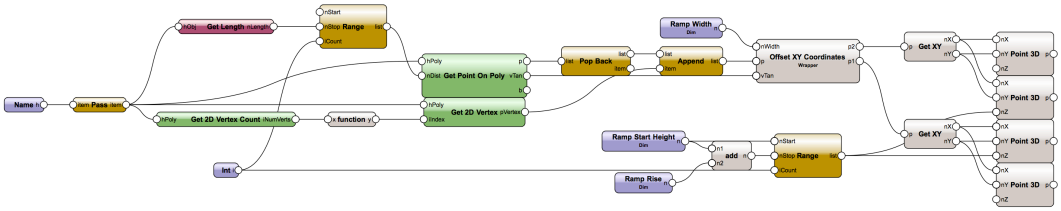


BILD 45 – Ein Netzwerk, das vier Listen mit 3D-Punkten erzeugt, die einen Abstand zur Polylinie aufweisen

Jetzt brauchen wir vier **Curve**-Nodes aus dem Ordner „Objects“ → „NURBS“ → „Curves“ – eine für jeden unserer Nodes **Point 3D** sowie einen **Real**-Node aus dem Ordner „Input“.

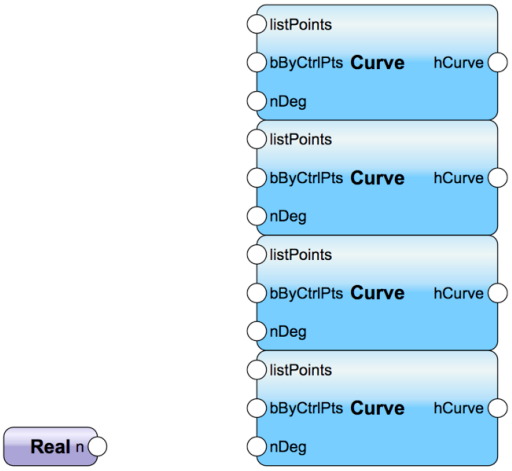


BILD 46 – Der **Real**-Node aus dem Ordner „Input“ und vier **Curve**-Nodes aus dem Ordner „NURBS“

# TUTORIAL 2: **MARIONETTE-OBJEKT ERZEUGEN** [FORTS.]

Geben Sie dem **Real**-Node in der Infopalette im Feld **real** den Wert 3 und verbinden Sie den Output mit dem Input „nDeg“ jedes der vier **Curve**-Nodes. Verbinden Sie dann die Outputs jedes der Nodes **Point 3D** mit den Inputs „listPoints“ jedes der **Curve**-Nodes. Dies erzeugt eine NURBS-Kurve für jede Liste mit 3D-Punkten mit einem Gradwert 3. Der Vorgabewert des Inputs „bByCtrlPts“ ist „Falsch“, was bedeutet, dass die Kurven mit interpolierten Punkten statt mit Kontrollpunkten gezeichnet werden.

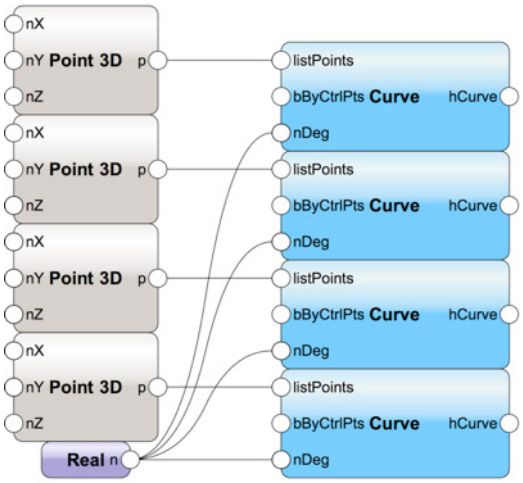


BILD 47 – Die Nodes **Point 3D** sind mit den **Curve**-Nodes verbunden

Als Nächstes benötigen wir den Node **Ordered List** aus dem Ordner „Data Flow“, einen **Group**-Node aus dem Ordner „Operations“ und den Node **Loft Surface** aus dem Ordner „Objekte“ → „NURBS“ → „Surface“.

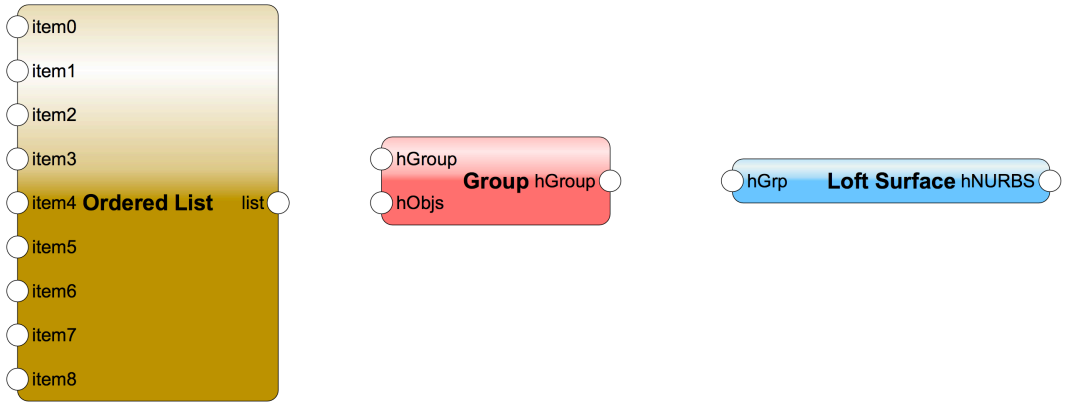


BILD 48 – Der Node **Ordered List** aus dem Ordner „Data Flow“, der **Group**-Node aus dem Ordner „Operations“ und der Node **Loft Surface** aus dem Ordner „NURBS“

# TUTORIAL 2: **MARIONETTE-OBJEKT ERZEUGEN** [FORTS.]

Der Node **Ordered List** wird dazu verwendet, eine Liste mit Kurven aus den vier **Curve**-Nodes zu erzeugen. Wenn wir den Input des Nodes **Loft Surface** betrachten, steht dort „groupCurves“, nicht „listCurves“, deshalb müssen wir zuerst unsere Kurvenliste mit dem **Group**-Node in eine Gruppe umwandeln. Um diese Nodes zum Netzwerk hinzuzufügen, verbinden Sie die Outputs der vier **Curve**-Nodes mit den ersten vier Inputs des Nodes **Ordered List**, dann den Output des Node **Ordered List** mit dem Input „hObjs“ des **Group**-Nodes und den Output des **Group**-Nodes mit dem Input des Nodes **Loft Surface**. Stellen Sie schließlich sicher, dass in der Infopalette des Nodes **Loft Surface** die Option **rule** aktiviert ist. Dies bedeutet, dass die Fläche an jeder Kurve einen Knick hat und keinen gekurvten Bogen durch alle vier macht.

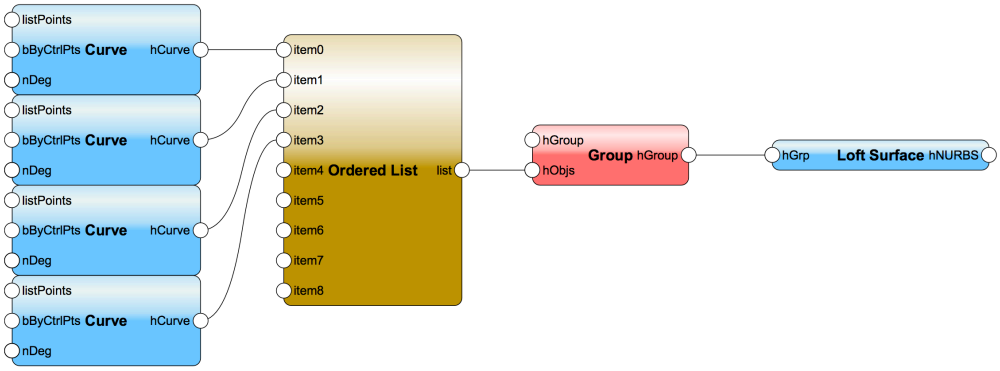


BILD 49 – Die **Curve**-Nodes sind mit den Nodes **Ordered List** verbunden, um eine hochgezogene Fläche zu erzeugen.

Das Netzwerk, das wir gebaut haben, erzeugt zu diesem Zeitpunkt eine hochgezogene Fläche in Form einer Rampe.

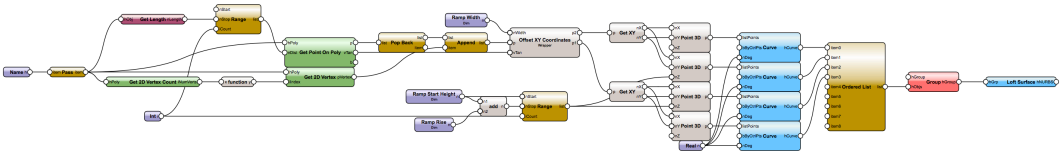

BILD 50 – Ein Netzwerk, das eine hochgezogene Fläche in Form einer Rampe erzeugt

# TUTORIAL 2: **MARIONETTE-OBJEKT ERZEUGEN** [FORTS.]

Erzeugen Sie die Rampe, indem Sie mit der rechten Maustaste auf einen Node im Netzwerk klicken und **Marionette Script ausführen** wählen. Wollen Sie die Rampe in einer 3D-Ansicht anzeigen, schalten Sie die Mehrfensteransicht ein (**Ansicht → Ansichtsfenster → Mehrere Ansichtsfenster** wählen oder auf  in der Darstellungsleiste klicken oder einfach die Taste „Q“ drücken). Node-Drähte können nur in der Ansicht „2D-Plan“ verbunden und abgelöst werden, deshalb müssen Sie mit Marionette in dieser Ansicht arbeiten, aber Sie können die 3D-Objekte in den anderen Ansichtsfenstern sehen.

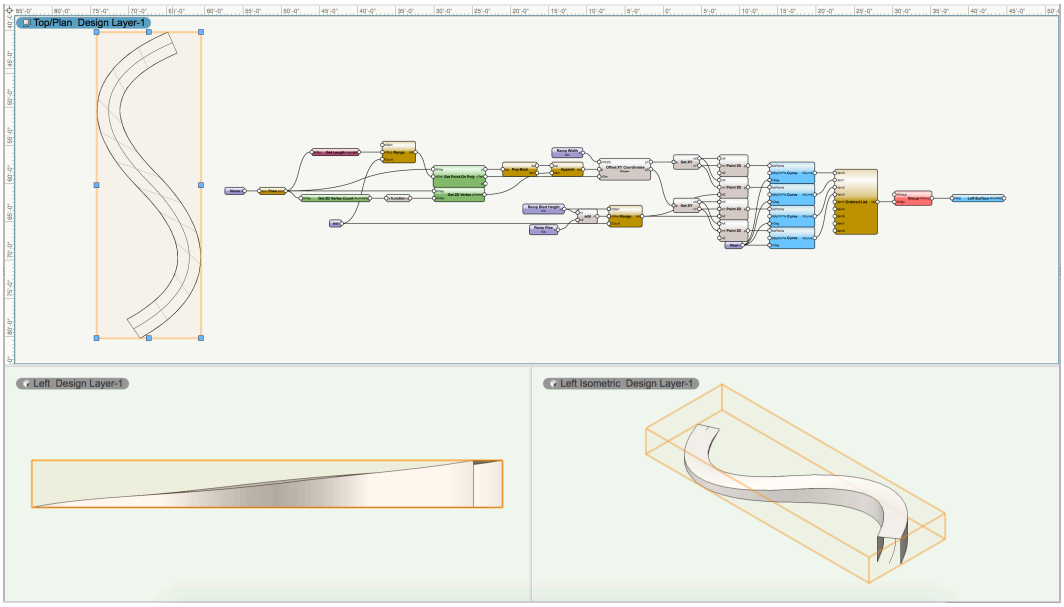


BILD 51 – Ein Marionette-Netzwerk, das eine Rampe erzeugt, angezeigt in mehreren Ansichtsfenstern

# TUTORIAL 2: **MARIONETTE-OBJEKT ERZEUGEN** [FORTS.]

Wir haben jetzt eine hochgezogene Rampe, die wir in einem größeren Modell verwenden können. Wenn wir das Netzwerk wrappen, erscheinen die benannten Nodes als Datenfelder in der Infopalette. Wrappen Sie das Netzwerk, indem Sie mit der rechten Maustaste auf einen Node klicken und **Wrap Marionette Network** wählen.


BILD 52 – Die Infopalette eines Wrappers zeigt die mit Namen versehenen **Input**-Nodes als Datenfelder.

# TUTORIAL 2: **MARIONETTE-OBJEKT ERZEUGEN** [FORTS.]

Jetzt, wo wir einen Wrapper haben, können wir einen Schritt weiter gehen und diesen in ein intelligentes Objekt umwandeln. Beachten Sie dabei aber, dass intelligente Objekte mit 3D-Geometrie in Vectorworks eine separate 2D-Geometrie benötigen, damit sie in einer 2D-Ansicht korrekt dargestellt werden. Dies ist sinnvoll bei Planzeichnungen, in die Darstellung von Objekten bestimmten Konventionen folgt, die nicht unbedingt der Ansicht eines 3D-Objekts von oben entsprechen. Bevor wir also diesen Wrapper in ein intelligentes Objekt umwandeln, müssen wir zuerst ein neues Netzwerk bauen, das eine 2D-Version der Rampe erzeugt und dieses in unser bestehendes Netzwerk einbauen. Wir verwenden die 2D-Koordinaten, die vom Wrapper **Offset XY Coordinates** generiert wurden, um ein geschlossenes 2D-Polygon zu erzeugen.

Zuerst packen wir das Netzwerk aus („unwrap"), um weiter daran zu arbeiten. (Wir könnten auch im Wrapper arbeiten, aber der Debug-Modus funktioniert nicht in einem Wrapper.) Packen Sie das Netzwerk aus, indem Sie mit der rechten Maustaste auf den Wrapper klicken und im Kontextmenü **Unwrap Marionette Network** wählen.

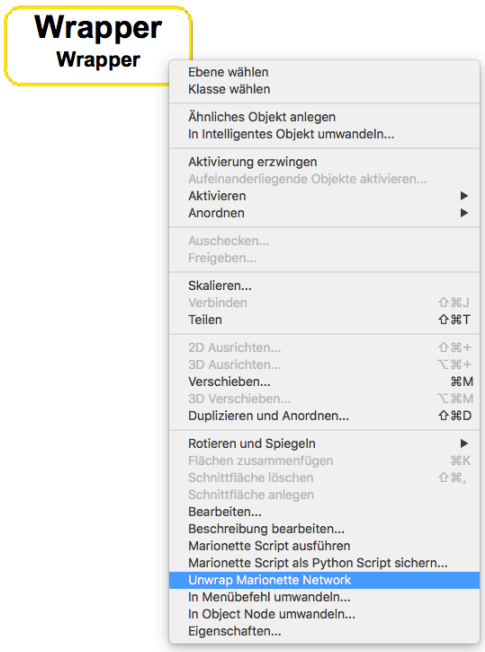


BILD 53 – Das Kontextmenü eines Marionette-Wrappers

Um ein neues Polygon aus den Koordinaten zu erzeugen, die wir schon haben, müssen wir die beiden Koordinatenlisten zu einer vereinigen. Wir benötigen einen **List**-Node und den Node **Reverse List**, beide aus dem Ordner „Data Flow".


BILD 54 – Der Node **Reverse List** und der **List**-Node aus dem Ordner „Data Flow"

# TUTORIAL 2: **MARIONETTE-OBJEKT ERZEUGEN** [FORTS.]

Damit wir die Koordinaten zum Zeichnen des Polygons in der richtigen Reihenfolge erhalten, müssen wir eine der Listen umdrehen. Verbinden Sie dazu den Output „p1“ des Wrappers **Offset XY Coordinates** mit dem Input „item1“ des **List**-Nodes und den Output „p2“ des Wrappers mit dem Input des Nodes **Reverse List**. Dann wird der Output des Nodes **Reverse List** mit dem Input „item2“ des **List**-Nodes verbunden.

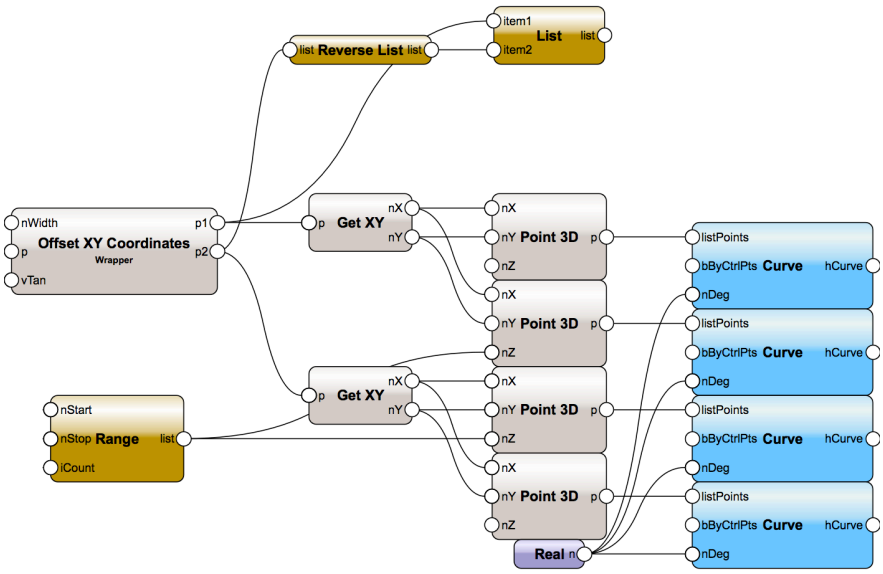


BILD 55 – Aus den Outputs des Wrappers **Offset XY Coordinates** eine Liste mit Punkten erzeugen

Jetzt haben wir die Punkteliste in der richtigen Reihenfolge und können den Node **Polygon 2D** aus dem Ordner „Objects“ → „Poly“ → „2D“ holen und mit dem Output des **List**-Nodes verbinden.

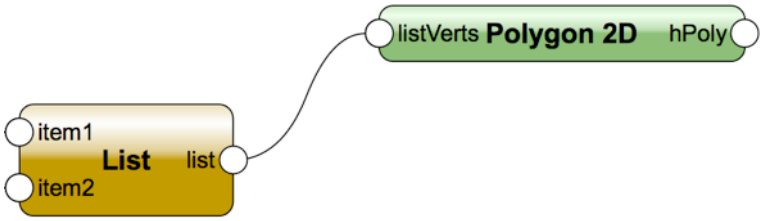


BILD 56 – Der **List**-Node wird mit dem Node **Polygon 2D** verbunden.

# TUTORIAL 2: **MARIONETTE-OBJEKT ERZEUGEN** [FORTS.]

Dies erzeugt ein Polygon mit allen Eckpunkten, aber wir müssen alle Punkte außer den Endpunkten der Rampe in Bézier-Scheitelpunkte umwandeln. Zu diesem Zweck erzeugen wir eine Index-Liste, die den Scheitelpunkten entspricht, die wir umwandeln wollen. Für die Liste benötigen wir den Node **Get List Length** und einen **Series**-Node.

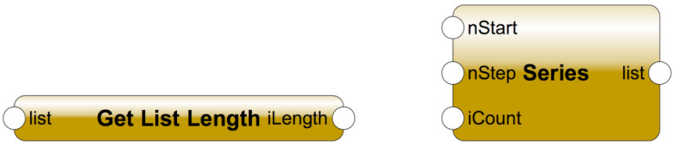


BILD 57 – Der Node **Get List Length** und der **Series**-Node aus dem Ordner „Data Flow"

Verbinden Sie den Output des **List**-Nodes mit dem Input des Nodes **Get List Length** und dann den Output des Nodes **Get List Length** mit dem Input „iCount" des **Series**-Nodes. Der **Series**-Node ähnelt dem **Range**-Node insofern, dass er eine Folge von Zahlen erzeugt, aber statt eines Anfangs, eines Endes und eines Zählers benötigt der **Series**-Node einen Anfang, einen Schritt und einen Zähler. Aus diesem Grund produziert der **Series**-Node bei einem vorgegebenen Zähler „20" 20 Werte, während der **Range**-Node 21 Werte produziert.

Sind der Node **Get List Length** und der **Series**-Node mit dem Netzwerk verbunden, verfügen wir über eine Index-Liste für alle Scheitelpunkte in unserem Polygon. Jetzt müssen wir die Indizes für die Scheitelpunkte finden, die wir nicht ändern wollen, und diese von der Liste entfernen. Wir wissen, dass die ersten und letzten Koordinaten der Outputs „p1" und „p2" Scheitelpunkte sind, die wir nicht ändern wollen, deshalb können wir diese mit einem weiteren Node **Get List Length** und zwei Nodes **User Function** finden. Da sich beide Nodes bereits in der Zeichnung befinden, können Sie diese einfach kopieren und einfügen. Bei den beiden Nodes **User Function** ist die Funktion dieselbe wie in den Originalen, deshalb ist es nicht notwendig, diese zu ändern. Wir brauchen auch einen weiteren **Integer**-Node sowie einen weiteren Node **Ordered List**. Diese können ebenfalls kopiert und eingefügt werden. Ändern Sie den Wert des neuen **Integer**-Nodes auf 0 und verbinden Sie ihn mit dem Input „item0" des Nodes **Ordered List**. Verbinden Sie jeden Node **Get List Length** mit den Nodes **User Function** und verbinden Sie diese und den zweiten Node **Get List Length** in der unten gezeigten Reihenfolge.

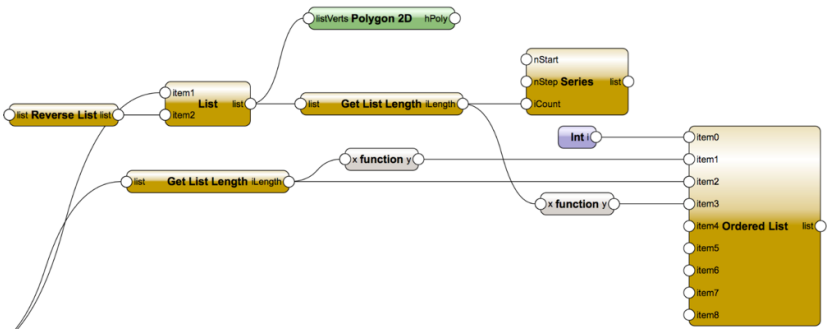


BILD 58 – Eine Liste von Indizes erzeugen, die entfernt werden

# TUTORIAL 2: **MARIONETTE-OBJEKT ERZEUGEN** [FORTS.]

Wenn Sie den Node **Dummy End** aus dem Ordner „Debug“ mit dem Output des Nodes **Ordered List** verbinden und im Debug-Modus auf den Draht klicken, sollten Sie die hier gezeigten vier Werte sehen:

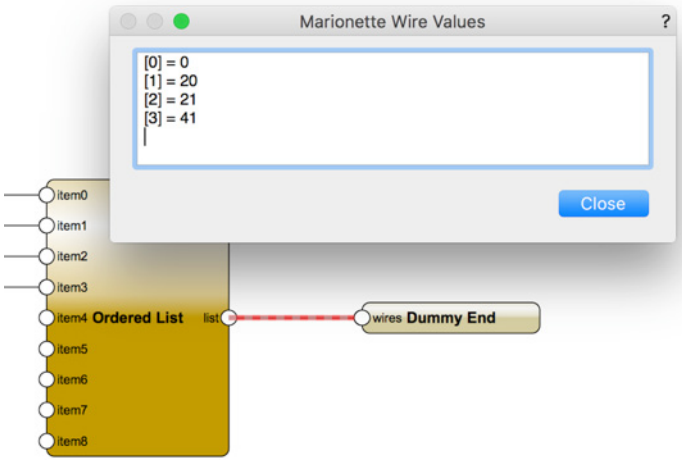


BILD 59 – Die vier Index-Werte, die entfernt werden sollen

Sind die vier Werte nicht in der richtigen Reihenfolge, versuchen Sie, die Drähte in eine andere Reihenfolge zu bringen. Sind die Werte nicht korrekt, stellen Sie sicher, dass der Wert im Datenfeld der beiden Nodes **User Function** „x-1“ ist. Haben Sie die korrekten Werte, können Sie den Node **Dummy End** wieder aus dem Netzwerk entfernen. Jetzt benötigen wir einen **Remove**-Node aus dem Ordner „Data Flow“.


BILD 60 – Der **Remove**-Node aus dem Ordner „Data Flow“

Verbinden Sie den Output des **Series**-Nodes mit dem Input „list“ des **Remove**-Nodes und verbinden Sie den Output des Nodes **Ordered List** mit dem Input „item“ des **Remove**-Nodes. Jetzt haben wir eine Index-Liste, die wir in Bézier-Scheitelpunkte umwandeln werden. Da wir unser Netzwerk im Debug-Modus ausgeführt haben, sollten wir in unserer Zeichnung ein 2D-Polygon mit allen Eckpunkten haben. Außerdem ist das Polygon nicht geschlossen, deshalb müssen wir es schließen und die Eckpunkte konvertieren. Dafür brauchen wir den Node **Set Closed 2D** und den Node **Change Vertex Type**, beide aus dem Ordner „Objects“ → „Poly“ → „2D“.

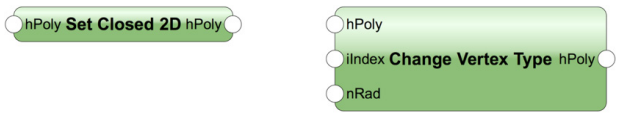


BILD 61 – Der Node **Set Closed 2D** und der Node **Change Vertex Type** aus dem Ordner „Objects“

# TUTORIAL 2: **MARIONETTE-OBJEKT ERZEUGEN** [FORTS.]

Aktivieren Sie zuerst den Node **Change Vertex Type** und wählen Sie im Einblendmenü **Vertex Type** „Bézier“. Dies wandelt die definierten Indizes in Bézier-Scheitelpunkte um. Verbinden Sie dann den Output des Nodes **Polygon 2D** mit dem Input des Nodes **Set Closed 2D** und verbinden Sie den Output des Nodes **Set Closed 2D** mit dem Input „hPoly“ des Nodes **Change Vertex Type**. Verbinden Sie schließlich den **Remove**-Node mit dem Input „iIndex“ des Nodes **Change Vertex Type**.

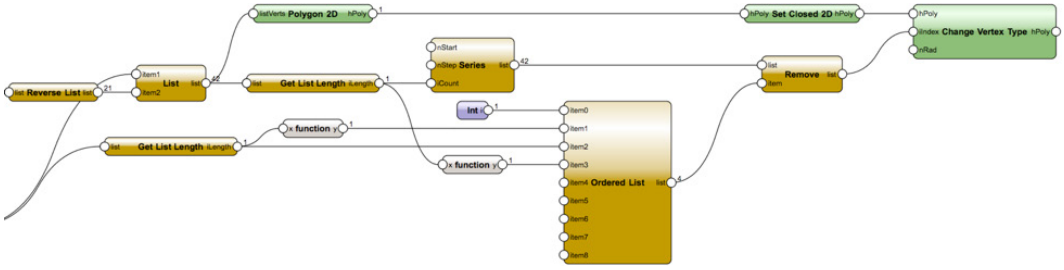


BILD 62 – Dieser Abschnitt des Netzwerks erzeugt ein 2D-Polygon in Form der Rampe

Im Bild oben zeigen die Zahlen rechts neben einigen der Outputs die Anzahl der Werte an, die durch die Drähte laufen. Diese werden angezeigt weil das Netzwerk zuletzt im Debug-Modus ausgeführt wurde.

Da dieses Polygon eine 2D-Repräsentation unserer gebogenen NURBS-Rampe ist, wollen wir seine Attribute getrennt von der eigentlichen Rampe definieren. Der einfachste Weg 2D-Attribute zu steuern ist über die Klasse, deshalb benötigen wir den Node **Set Class** aus dem Ordner „Attributes“ → „Set Operations - Attributes“ sowie einen **String**-Node aus dem Ordner „Inputs“ → „Basic“. Der **String**-Node kann dazu verwendet werden den Namen der Klasse, die Sie wählen wollen, einzugeben. Geben Sie im Textfeld **string** in der Infopalette den Vorgabewert „Keine“ ein. Dieses Textfeld des **String**-Nodes muss in der Infopalette unseres Marionette-Objekts sichtbar sein. Geben Sie dem Node deshalb den Namen „Klasse Rampe 2D“.


BILD 63 – Der **String**-Node aus dem Ordner „Inputs“ und der Node **Set Class** aus dem Ordner „Attributes“

# TUTORIAL 2: **MARIONETTE-OBJEKT ERZEUGEN** [FORTS.]

Verbinden Sie den Output des Nodes **Change Vertex Type** mit dem Input des Nodes **Set Class** und verbinden Sie den Output des **String**-Nodes mit dem Input „sClass“ des Nodes **Set Class**. Unser Netzwerk ist fast fertig.

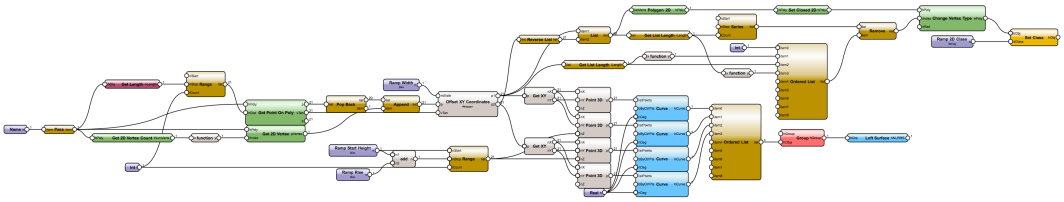

BILD 64 – Ein Netzwerk, das eine NURBS-Rampe und ein 2D-Polygon erzeugt

Wenn wir die Reihenfolge der Datenfelder in der Infopalette unseres Marionette-Objekts festlegen wollen, können wir dies tun, indem wir sie nummerieren. Ein **Input**-Node wird nummeriert, indem Sie einfach eine Zahl vor dem Namen des Nodes eingeben. Die Zahl wird in der Infopalette nicht angezeigt.


BILD 65 – **Input**-Nodes können eine Zahl vor ihrem Namen erhalten, damit Sie in der Infopalette des Marionette-Objekts in der richtigen Reihenfolge erscheinen

Das Letzte, das wir tun müssen, bevor wir das Netzwerk in ein Marionette-Objekt umwandeln, ist den **Name**-Node, den wir am Anfang des Netzwerks verwendet haben, gegen den Node **Control Geometry** aus dem Ordner „Input“ auszutauschen.


BILD 66 – Der Node **Control Geometry** aus dem Ordner „Input“

Wie bereits erwähnt, wird der **Name**-Node in Netzwerken und Wrappern verwendet, wenn auf bestehende Geometrie Bezug genommen wird, aber in einem Marionette-Objekt wird der Node **Control Geometry** verwendet. Tauschen Sie die Nodes aus, indem Sie den **Name**-Node löschen oder ablösen und den Node **Control Geometry** mit dem **Pass**-Node verbinden.


BILD 67 – Den **Name**-Node durch den Node **Control Geometry** ersetzen

# TUTORIAL 2: **MARIONETTE-OBJEKT ERZEUGEN** [FORTS.]

Jetzt kann das Netzwerk in ein Marionette-Objekt umgewandelt werden. Wrappen Sie das Netzwerk, indem Sie mit der rechten Maustaste auf einen Node im Netzwerk klicken und **Wrap Marionette Network** wählen. Geben Sie, wenn gewünscht, einen Namen und eine Beschreibung ein. Aktivieren Sie dann den ursprünglichen Polylinien-Pfad zusammen mit dem Wrapper, klicken Sie mit der rechten Maustaste auf den Wrapper und wählen Sie **In Object Node umwandeln**.

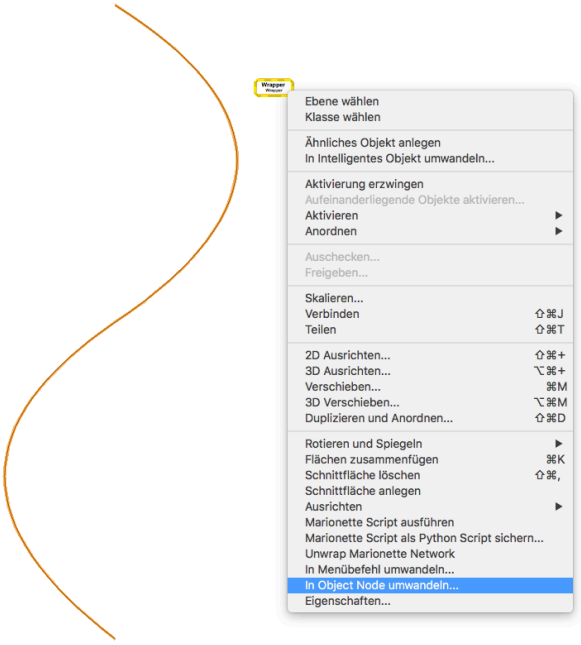


BILD 68 – Einen Wrapper und Kontrollgeometrie in ein Marionette-Objekt umwandeln

Schalten Sie im erscheinenden Dialogfenster **Reset beim Verschieben** aus und klicken Sie auf **OK**. Sie sollten jetzt eine Rampe haben, die Sie in einem 3D-Modell verwenden und in einer 2D-Zeichnung anzeigen können.

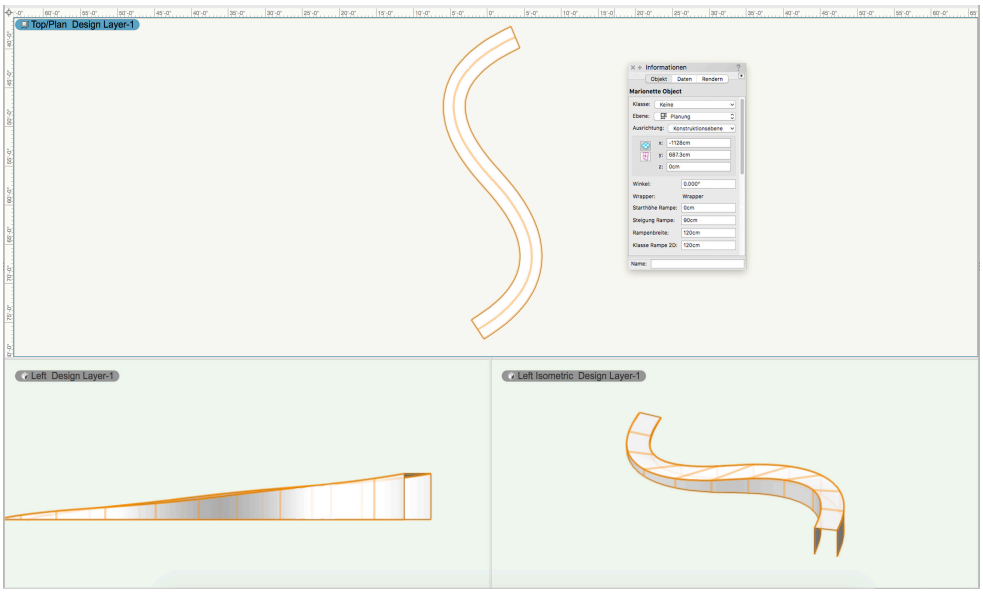


BILD 69 – Drei Ansichten der gebogenen Rampe

# KONTAKTIEREN SIE UNS

bei Interesse an Schulungen oder für Feedback.

E-Mail an:

**info@computerworks.de** bzw. **info@computerworks.ch**.

**Deutschland & Österreich**
ComputerWorks GmbH
Schwarzwaldstraße 67
79539 Lörrach
www.computerworks.de

**Schweiz**
ComputerWorks AG
Florenz-Straße 1e
4142 Münchenstein
www.computerworks.ch

ComputerWorks

AUTHORIZED DISTRIBUTOR