

*Diese Einführung beschreibt ausschließlich die Programmierung von Pd mit nachrichtenverarbeitenden Objekten. Audioobjekte werden nicht angesprochen, ebenso wenig Datenstrukturen, Sequencing, Graphik und Objekte zum Speichern und Bearbeiten von Tabellen.*

## **1. Grundlagen**

**Puredata** (Pd) ist ein Programm zur Erstellung und Ausführung von Echtzeit-Computermusik-Anwendungen mit Erweiterungen für Computergraphik und Video. Hauptautor ist Miller Stuart Puckette (UCSD), der ab 1986 am IRCAM das Programm Max zur Steuerung spezieller DSP-Hardware vom Macintosh aus entwickelt hatte. Nach seinem Ausscheiden aus dem IRCAM begann er, 1996 mit Puredata ein verbessertes System zu entwickeln; eine Variante seines Max wurde verkauft und als Max/MSP kommerziell weiterentwickelt und vertrieben.

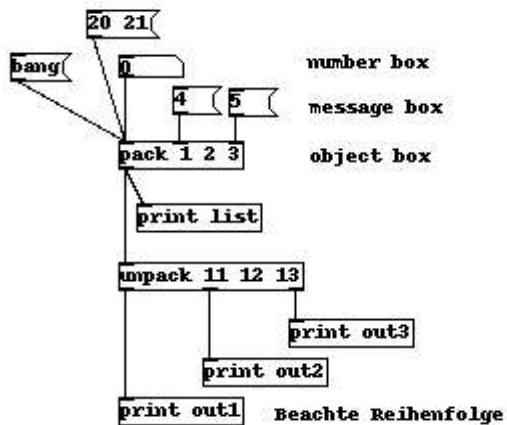
Puredata ist **freie Software**, d.h. die Lizenz erlaubt es dem Benutzer, den frei verfügbaren Quellcode zu analysieren, weiterzuentwickeln und zu verbreiten, solange dieser das Programm unter der gleichen Lizenz belässt. Man findet daher eine aktive Entwickler- und Nutzergemeinde, die viele Erweiterungspakete und Materialien frei zur Verfügung stellt. Pd läuft auf Irix, Linux, Mac OS X und Windows, aktueller Versionsstand der offiziellen Ausgabe ist 0.40-2; die erweiterte Ausgabe mit vielen freien Zusätzen ist in Version 0.39-2 erhältlich.

Pd-Programme beschreiben einen Datenfluss; die Erstellung und Darstellung des **Datenflussdiagramms** erfolgt graphisch. Dieses ähnelt der Verkabelung analoger Synthesizer (daher die Verwendung des Ausdruckes "**Patch**" für das Pd-Programm) oder dem Entwurf digitaler Schaltungen. Mehrere Patches können gleichzeitig aktiv sein. Im **Editiermodus**, den man am einfachsten mittels Ctl-e betritt und verlässt, werden Objekte und Verbindungen erzeugt (Put-Menü) und bearbeitet. Auch im Editiermodus bleibt das Programm (soweit möglich) aktiv. Rechtsklick auf ein Objekt führt zu den Objekteigenschaften und zum Hilfepatch des Objekts.

Pd-Patches bestehen aus **Objekten**, die über unidirektionale **Verbindungen** periodische **Signale** (etwa Audiodaten) und sporadische **Nachrichten** empfangen und versenden. Letztere können atomar (nicht teilbar) oder zusammengesetzt sein. Atomare Nachrichten sind Symbole, Zeiger (auf Datenstrukturen), Zahlen (wobei Pd hier nur mit Fließkommazahlen rechnet) sowie "bang" als spezielle Nachricht zum Triggern von Vorgängen ohne Übermittlung eines Wertes. Zusammengesetzte Nachrichten, die mit einer Zahl beginnen, heißen **Listen**; alle anderen dienen zum Ändern des Zustands eines angesprochenen Objekts. Listen können mit [pack] aus Atomen erzeugt und mit [unpack] wieder zerlegt werden. Die neueren Pd-Versionen kennen einige weitere listenverarbeitende Objekte.

Bei Objekten mit mehreren **Eingängen** führt meist nur der linke Eingang zur Aussendung einer Nachricht, während die anderen Eingänge lediglich Parameter setzen. Sendet man anstelle einer atomaren Nachricht eine Liste an das Objekt, so werden die Atome der Liste auf die Eingänge verteilt.

Bei Objekten mit mehreren **Ausgängen** erscheinen die Nachrichten am rechten Ausgang zuerst, dann nach links fortschreitend.



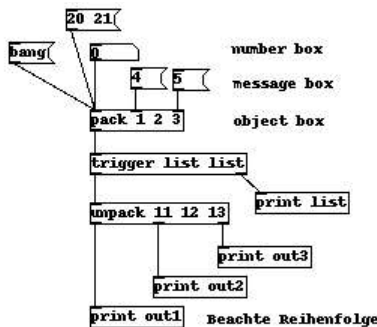
*Beispielpatch einf1.pd*

[pack], [unpack] mit Defaultwerten;

Nur Messages in den linken Eingang bewirken eine ausgehende Nachricht; Messages in nichtlinke Eingänge ersetzen Defaultwerte und bleiben gespeichert. Die Liste setzt die ersten beiden Eingänge.

Ausgänge werden von rechts nach links aktiviert.

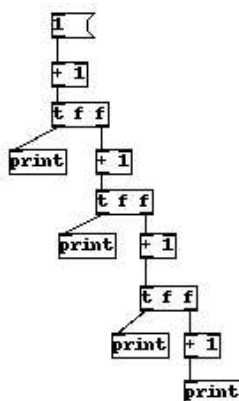
Ist ein Ausgang mit mehreren Eingängen verbunden, so werden die Verbindungen in der Reihenfolge ihrer Erzeugung bedient (im Gegensatz zu Max/MSP !). Beim Ersetzen von Objekten wird diese Reihenfolge nicht sicher bewahrt; man benutze deshalb immer das **[trigger]**-Objekt, um die Reihenfolge der Nachrichten festzulegen! [trigger] erlaubt in gewissem Maße auch Typkonversionen.



*Beispielpatch einf2.pd*

Das [trigger]-Objekt sorgt für vorhersehbare Abfolge der Nachrichten

Die Nachrichtenkaskade bewegt sich zuerst in die Tiefe (**depth first message passing**)!



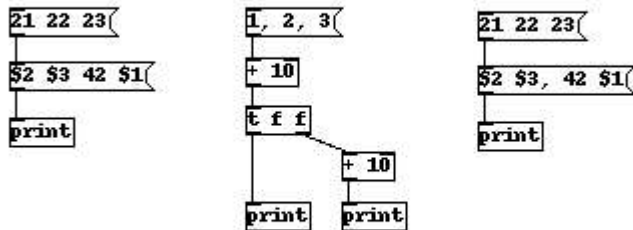
*Beispielpatch einf3.pd*

Beachte die Reihenfolge der Ausgaben!

## 2. Nachrichten

Die Message Box dient zum Versenden und Bearbeiten von Nachrichten. Atome eingehender Nachrichten können mit \$1, \$2, .. angesprochen und verarbeitet werden.

Durch Komma getrennte Nachrichten werden nacheinander gesendet, d.h. die nächste Nachricht wird gesendet, wenn die vorige in der Tiefe nicht mehr weiter kommt.



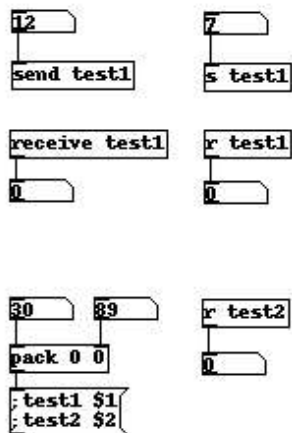
Beispielpatch einf4.pd

Variablensubstitution: Message-Box kann eingehende Werte über \$ ansprechen und eine neue Nachricht zusammenfügen.

Durch Komma getrennte Werte werden nacheinander ausgegeben, auch mit Variablensubstitution.

An Stelle direkter Verbindungen können Nachrichten auch über ein [send Name]-Objekt zu einem gleichnamigen [receive Name]-Objekt übermittelt werden. [netsend], [netreceive] bieten Kommunikation im LAN.

Beginnt eine Nachricht in einer Message Box mit Semikolon+Name, so wird sie an das benannte [receive]-Objekt versendet.



Beispielpatch einf5.pd

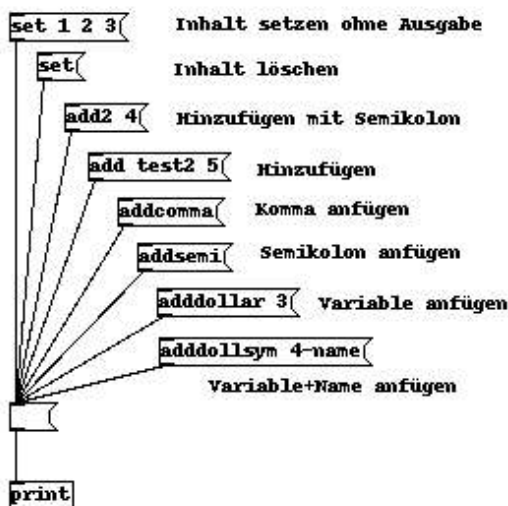
[send] sendet an alle [receive]-Objekte mit gleichem Namen.

Semikolon+Name in der Message Box sendet an alle [receive]-Objekte dieses Namens.

Die Message Box versteht Nachrichten zum Zusammensetzen und Erweitern von Nachrichten.

Über ein Sendpanel (zu erreichen über File -> Message) kann der Nutzer beliebige Nachrichten injizieren.





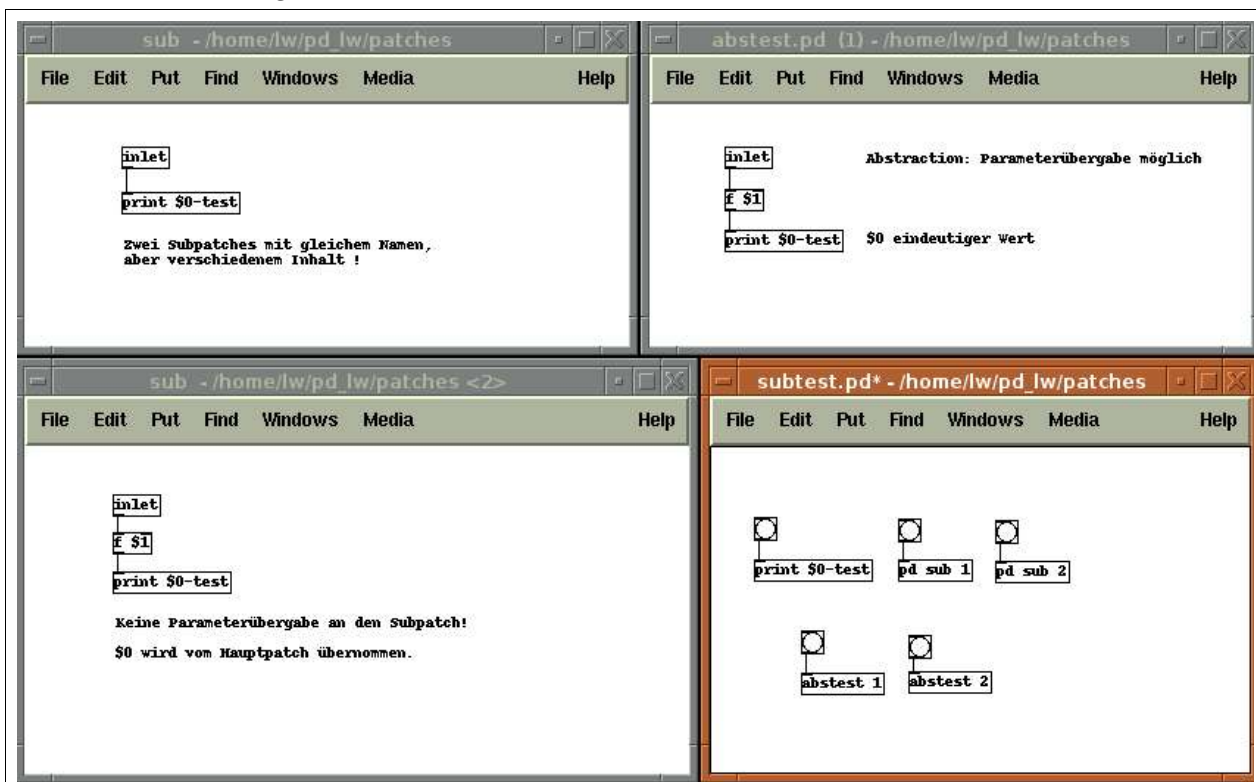
Beispielpatch einf6.pd

Komponieren von Nachrichten durch Messages an die Message Box

### 3. Subpatches. Abstractions.

Pd-Programme können zwei Arten von Unterprogrammen aufrufen. **Subpatches**, die mit [pd Name] erzeugt werden, erhöhen lediglich die Lesbarkeit. Sie sind Teil des Hauptprogramms und werden mit diesem abgespeichert. [inlet]- und [outlet]-Objekte bilden die Schnittstelle zum aufrufenden Programm, wo sie entsprechend ihrer topographischen Lage im Unterprogramm auftreten.

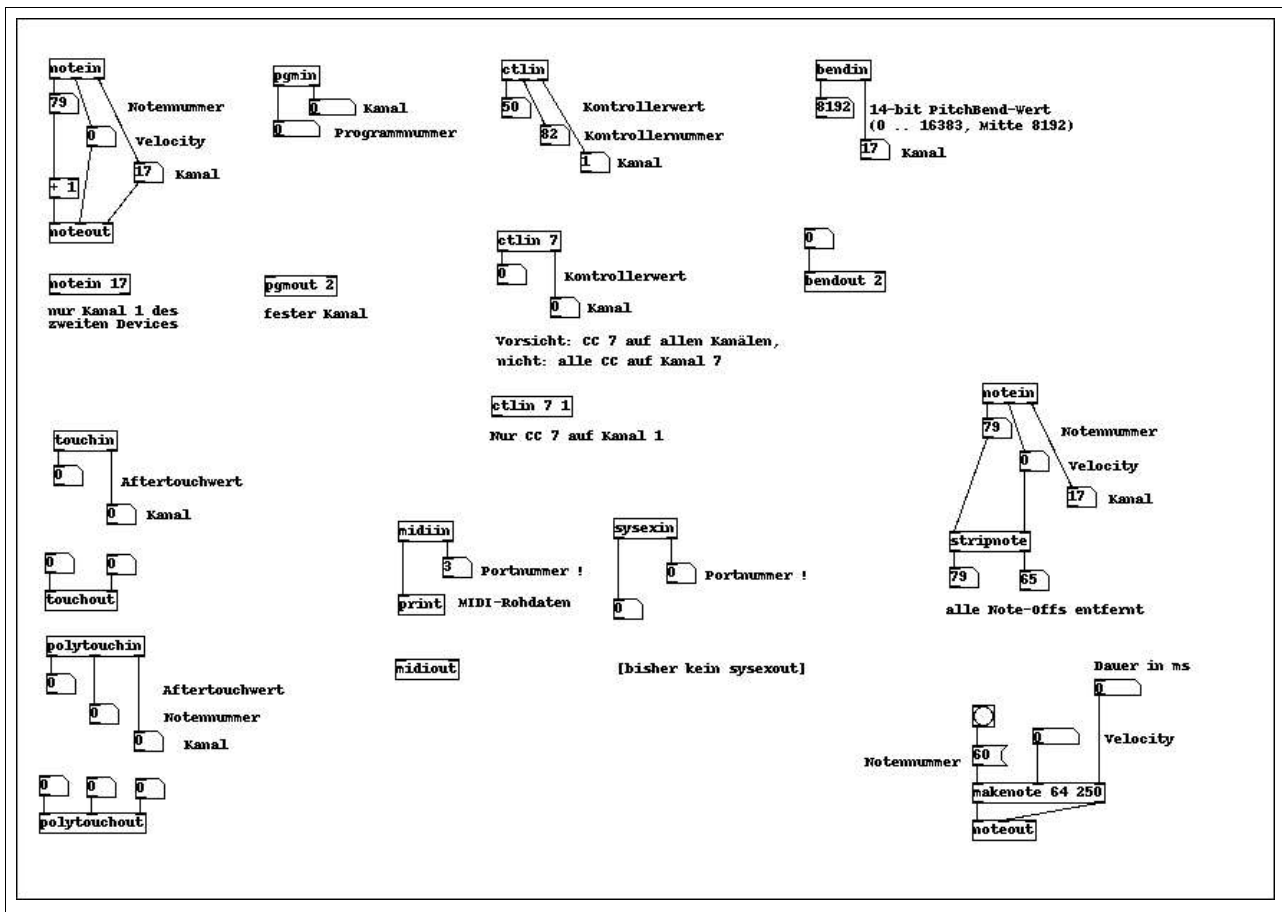
**Abstractions** dagegen liegen als separate Dateien im Filesystem und werden mit ihrem Namen aufgerufen wie eingebaute Objekte. Die Aufrufparameter (Zahlen oder Symbole) von Abstractions sind in Objektboxen als \$1, \$2, .. zugänglich. In Message-Boxen behalten \$1, \$2, .. die Bedeutung von Variablen!



[send]- und [receive]-Namen sind global, also werden Nachrichten an eine receive-Adresse eventuell an mehrere Instanzen einer Abstraktion oder auch zwischen diesen übermittelt. Falls die Instanzen mit unterschiedlichen Parametern aufgerufen wurden, lässt sich dies durch Voranstellen von \$1,.. verhindern. Auch unabhängige, gleichzeitig geöffnete Patches liegen im gleichen Namensraum! Voranstellen von \$0- erzeugt in jedem Fall lokale Namen.

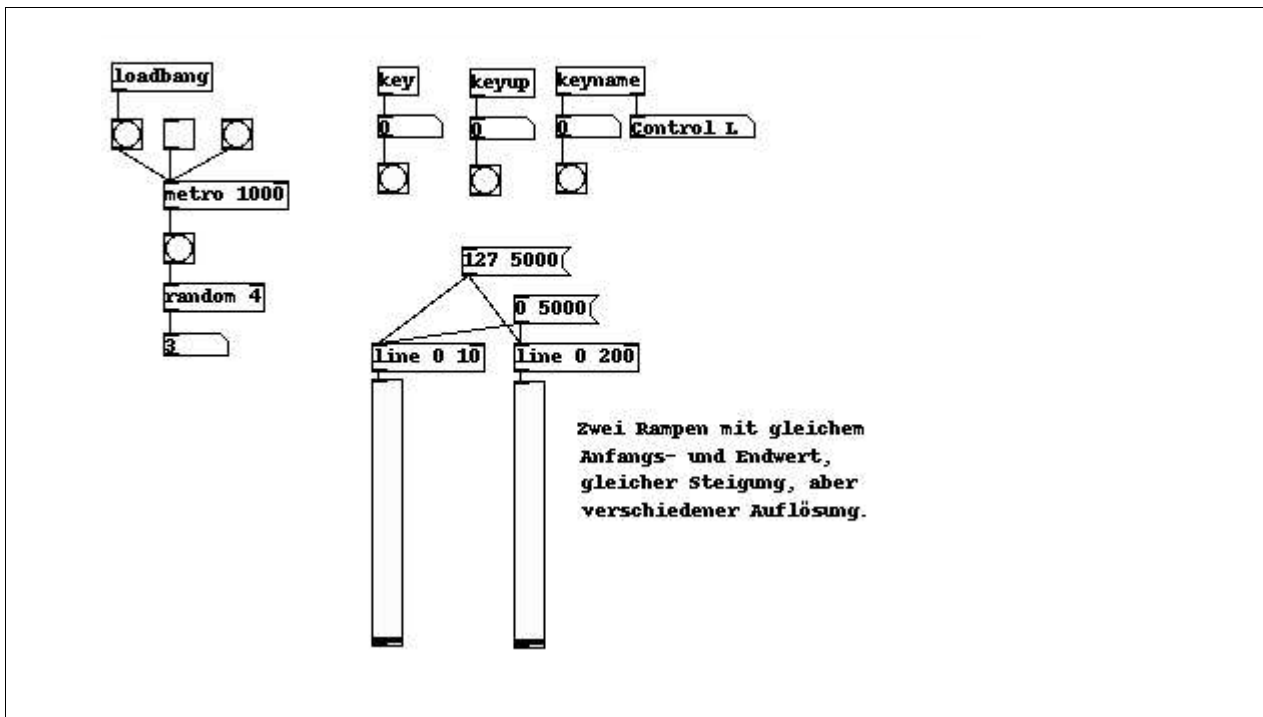
#### 4. MIDI-Ein-/Ausgabe

Eine Vielzahl von Objekten erlaubt die Ein- und Ausgabe von MIDI-Nachrichten. [stripnote] entfernt Note-On-Nachrichten mit Velocity 0 aus dem Datenstrom, [makenote] erzeugt eine Note gegebener Dauer.



#### 5. Sonstige Nachrichtenquellen

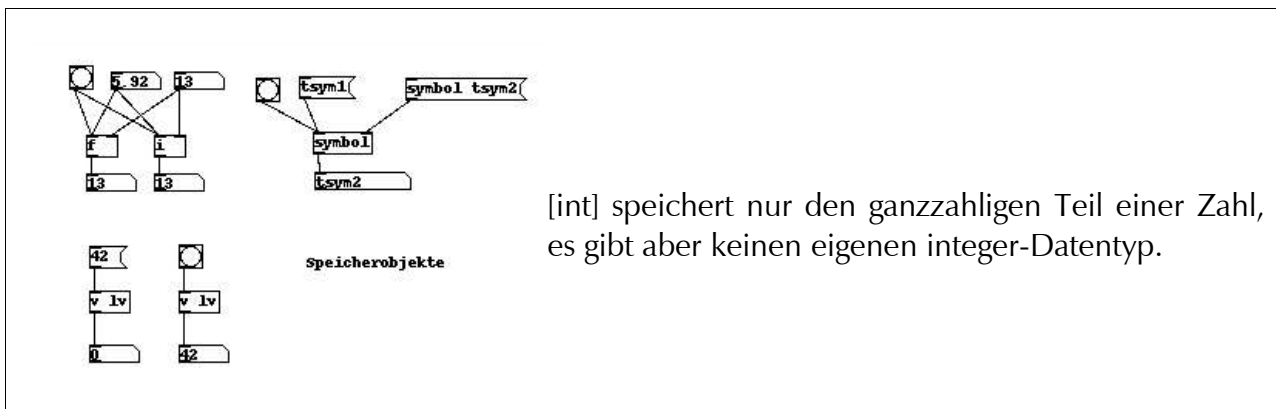
- [loadbang] sendet einen bang beim Programmstart und dient so zur Initialisierung von Programmen.
- [metro] erzeugt bangs in wählbarem zeitlichen Abstand (in ms angegeben); das Objekt wird durch einen bang oder eine 1 gestartet und durch 0 gestoppt.
- [random] erzeugt Pseudozufallszahlen in einem wählbaren Bereich, der als Parameter oder in den rechten Eingang eingegeben wird. (Parameter n ergibt die Zahlen 0,..,n-1.)
- [key] gibt den Code einer gedrückten Taste, [keyup] den Code einer losgelassenen Taste, [keyname] zeigt den Tastennamen sowie 1/0 für gedrückt/losgelassen.
- [line] erzeugt eine Zahlenrampe mit wählbarem Anstieg, Ziel und Auflösung. Die Parameter geben den Startwert und die Auflösung in ms an, der rechte Eingang empfängt die Dauer der Rampe, der linke den Endwert. Beide können wie üblich als Liste übergeben werden.



## 6. Speicherobjekte. Mathematisches und Logisches.

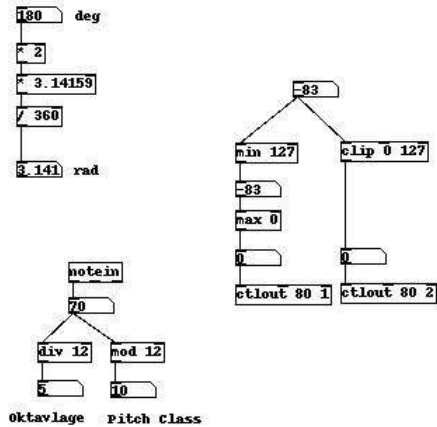
[float], [symbol], [int] **speichern** Nachrichten des jeweiligen Typs. Eingaben am rechten Eingang bewirken nur Setzen des Wertes ohne Ausgabe, [bang( am linken Eingang bewirkt Ausgabe des gespeicherten Wertes. Ein neuer Wert am linken Eingang überschreibt den Wert und gibt ihn gleichzeitig aus. Die Speicherobjekte können vorbelegt werden. Die häufig verfügbare Erweiterung "zexy" bietet mit [lister] auch ein Speicherobjekt für Listen.

[value name] definiert eine **global** zugängliche Variable. Eingabe einer Zahl speichert diese ohne Ausgabe, [bang( auf ein [value]-Objekt gleichen Namens liest sie aus, auch in Subpatches und gleichzeitig geöffneten Patches.



**Arithmetische** Objekte führen Operationen auf Zahlen aus:

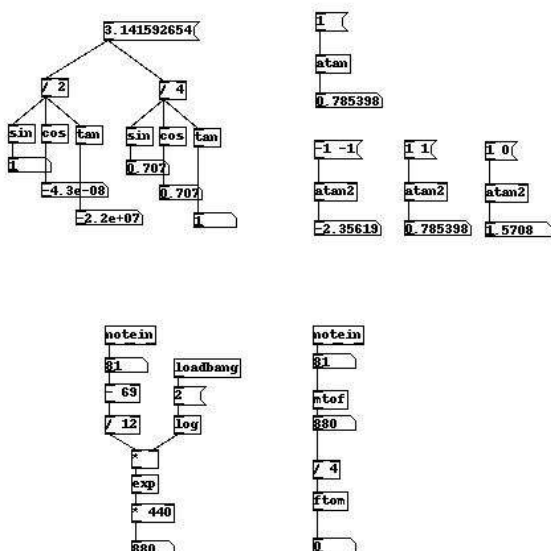
- [+], [-], [\*], [/] ergeben die Grundrechenarten.
- [div] liefert den Quotienten der ganzzahligen Division, [mod] oder [%] den Rest.
- [min] vergleicht den linken Eingang mit dem rechten oder dem vorbelegten Wert und sendet den kleineren von beiden, [max] den größeren. [clip] kombiniert beides.
- [abs] ergibt den Absolutwert der eingehenden Zahl.



Nicht ernst zu nehmendes Beispiel für die Konversion Gradmaß -> Bogenmaß, Ermittlung der Oktav- und Pitchklassennummer, Beschränken einer Ausgabe auf sinnvolle Werte.

### Höhere Mathematik bieten:

- [sin], [cos], [tan] berechnen Sinus, Cosinus und Tangens eines in radian (!) vorliegenden Eingangswertes .
- [atan] liefert den Arcustangens des Eingangswertes im Bereich  $-\pi/2 .. +\pi/2$ .
- [atan2] berechnet den Arcustangens des Quotienten seiner zwei Eingänge y (links) und x (rechts) (bei Versionen vor 0.38 vertauscht !). Die Vorzeichen der Eingänge werden ausgewertet, die Ausgabe liegt daher im Bereich  $-\pi .. +\pi$ . Sie stellt den Winkel (in radian) zwischen der positiven x-Achse und der Geraden vom Nullpunkt zum Punkt (x,y) dar.
- [pow y] erhebt den Eingangswert zur y-ten Potenz. Dabei darf y negativ sein, der Eingangswert nicht.
- [log] und [exp] stehen für natürlichen Logarithmus (zur Basis e) und Exponentialfunktion  $e^x$ .
- [sqrt] berechnet die Quadratwurzel.



Trigonometrische Funktionen erwarten Argumente in rad.

Arcustangens liefert Ergebnis in rad.

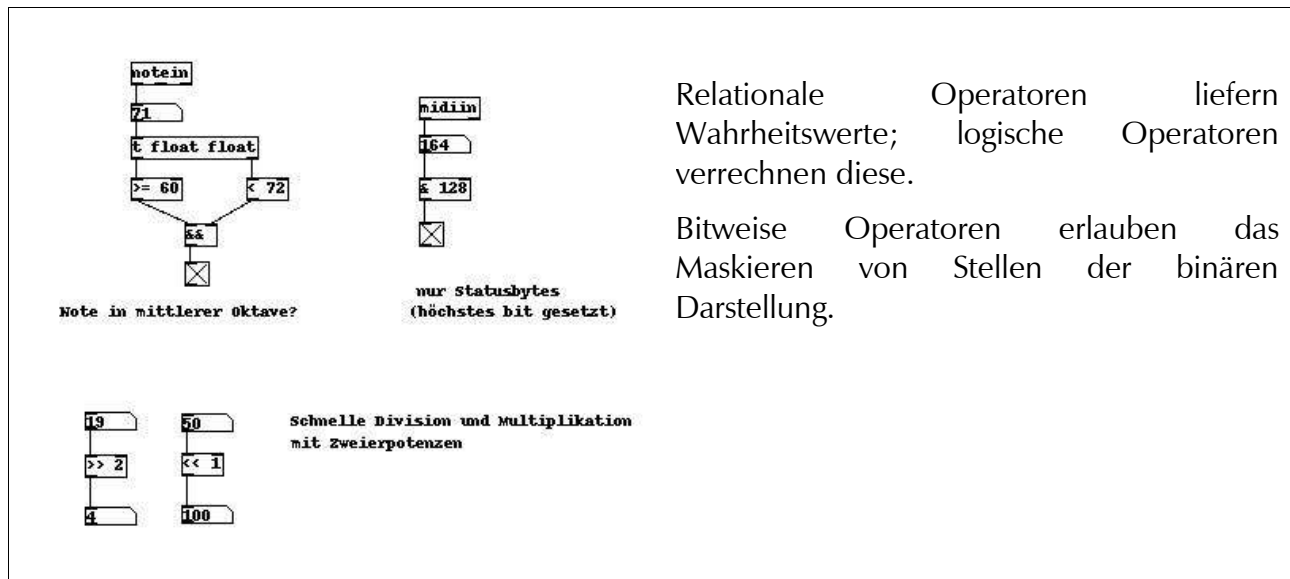
[atan2] liefert korrekte Winkel in allen vier Quadranten.

Konversion MIDI-Notennummer zu Frequenz "von Hand" oder mit Spezialobjekt [mtof].

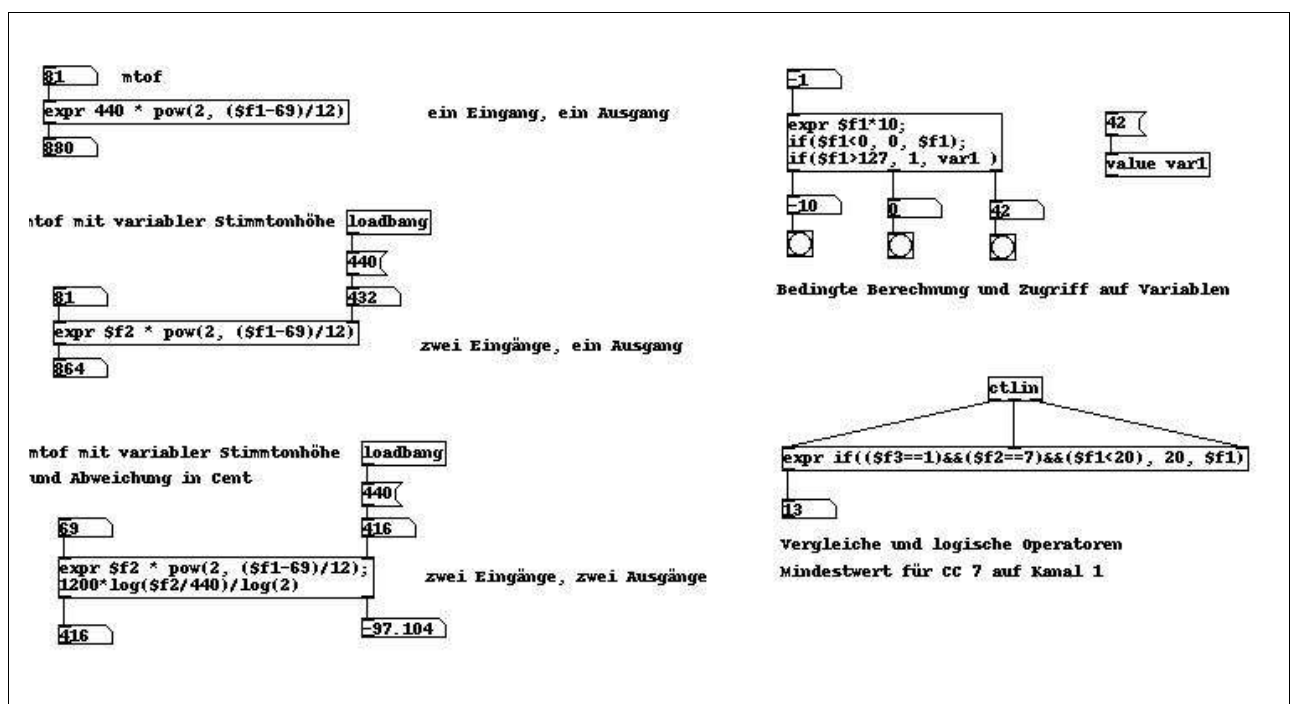
**Vergleichsobjekte** erzeugen die Wahrheitswerte 0 für falsch, 1 für wahr. [ $<$ ], [ $>$ ], [ $<=$ ], [ $>=$ ], [ $=$ ], [ $!=$ ].

Die **logischen** Operatoren [ $\&\&$ ] (UND) und [ $||$ ] (ODER) verknüpfen zwei Wahrheitswerte (0 für falsch, beliebige andere ganze Zahl für wahr) und liefern einen Wahrheitswert (0 für falsch, 1 für wahr).

Die **bitweisen** Operatoren verknüpfen die einzelnen Stellen der binären Darstellung der Eingangswerte [ $\&$ ] (bitweises UND), [ $||$ ] (bitweises ODER). [ $<<$ ] und [ $>>$ ] verschieben die binäre Darstellung um die angegebene Zahl von Stellen nach links bzw. rechts. Die neu erzeugten Stellen werden mit Nullen aufgefüllt, die überlaufenden Stellen sind verloren.



Das **[expr]-Objekt** bietet sich für komplexere Ausdrücke an. Alle angegebenen mathematischen, relationalen, logischen und bitweisen Funktionen können zu einem Ausdruck mit mehreren Eingangsvariablen kombiniert werden; [expr] zeigt dann entsprechend viele Eingänge. Mehrere durch Semikolon getrennte Ausdrücke erzeugen mehrere Ausgänge. Auch bedingte Berechnungen sind möglich, führen aber stets zu Ausgaben.

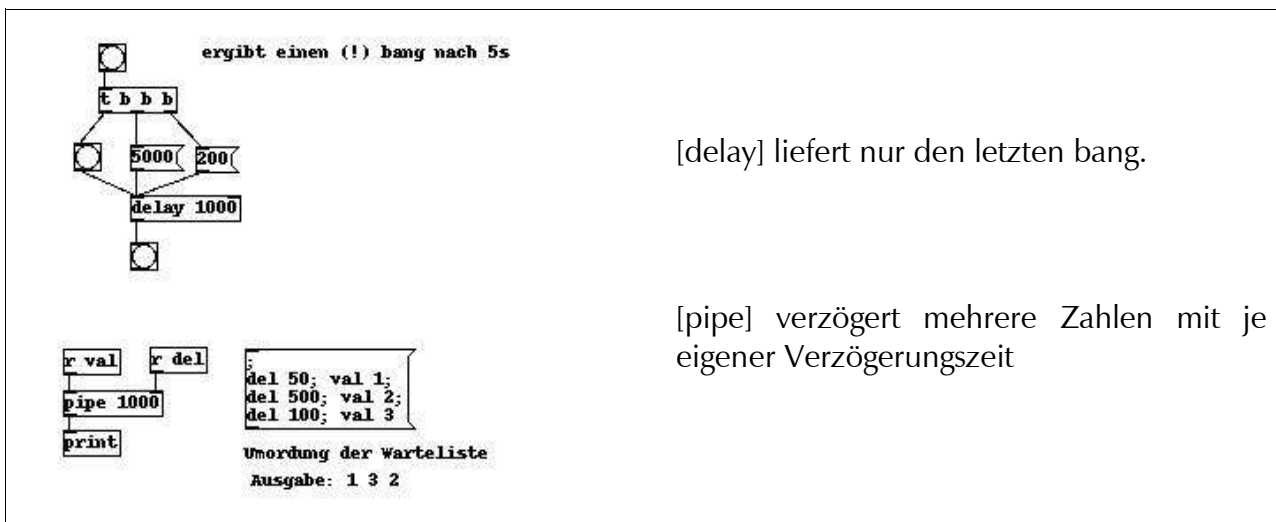




## 7. Verzögerung

[delay] gibt nach einer in ms als Parameter oder in den rechten Eingang eingegebenen Zeit einen bang aus. Eingabe einer Zahl in den linken Eingang überschreibt die Zeitdauer und startet die Verzögerung. Nachstarten des Delays innerhalb der Verzögerungsdauer löscht das verzögerte bang und startet einen neuen Delay.

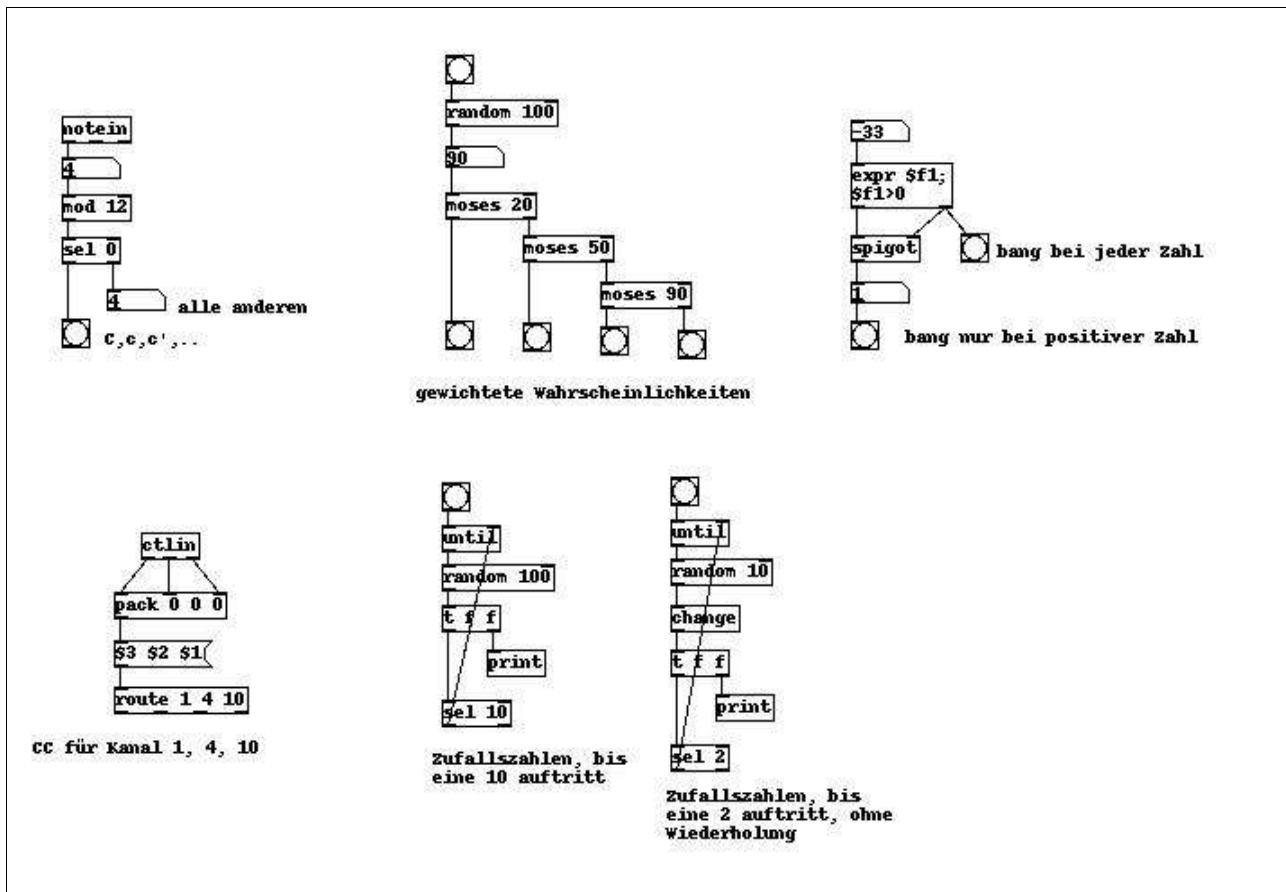
[pipe] dient der Verzögerung mehrerer Eingangsnachrichten (Zahlen oder Symbole je nach Aufrufparameter) um eine in ms gewählte Zeit. Wird eine neue Verzögerungszeit eingegeben, während noch Nachrichten in der Pipe sind, werden die Nachrichten umsortiert, d.h. jede behält ihre Verzögerungsdauer.



## 8. Verzweigungen

Verschiedene Objekte erlauben Programmverzweigungen je nach Wert der eingehenden Nachricht.

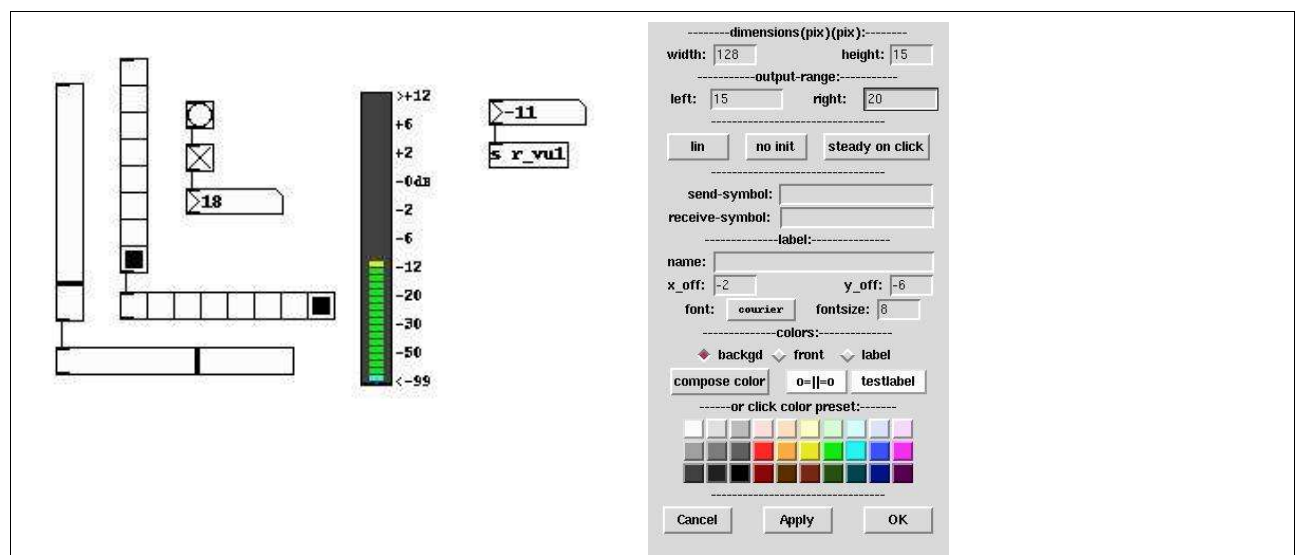
- [select] vergleicht eine eingehende Zahl oder ein Symbol mit einem oder mehreren vorgegebenen Werten. Ist nur ein Vergleichswert vorgegeben, kann dieser über einen rechten Eingang verändert werden. Für jeden Vergleichswert ist ein Ausgang vorhanden, der ein bang (nicht die eingehende Nachricht !) sendet, wenn der Eingangswert gleich dem Vergleichswert ist. Trifft kein Vergleich zu, wird die eingehende Nachricht über den rechten Ausgang weitergeleitet.
- [route] vergleicht das erste Atom einer eingehenden Nachricht mit seinen Aufrufparametern und leitet den Rest der Nachricht an den jeweiligen Ausgang weiter. Am rechten Ausgang erscheinen Nachrichten, die keinen Vergleich erfüllten. [route] kann nach Zahlen, Symbolen oder Datentypen sortieren.
- [moses] vergleicht eine eingehende Zahl mit seinem Aufrufparameter oder der rechts eingegebenen Zahl und gibt sie links aus, wenn sie kleiner als der Parameter ist, sonst rechts.
- [change] speichert eine eingehende Zahl und erzeugt nur dann einen Ausgang, wenn der Eingang vom gespeicherten Wert abweicht.
- [until] dient der Bildung von Schleifen. Eine Zahl am linken Eingang sendet die entsprechende Anzahl von bangs (for-Schleife). [bang( in den linken Eingang sendet bangs, bis ein [bang( den rechten Eingang erreicht oder der Rechner neu gestartet wird (until-Schleife). Im Fall einer Endlosschleife muss der pd-Prozess beendet werden, Schließen des Patches reicht nicht.



- [spigot] leitet den Datenstrom weiter, falls keine 0 am rechten Eingang anliegt -> ermöglicht if-Konstrukte.

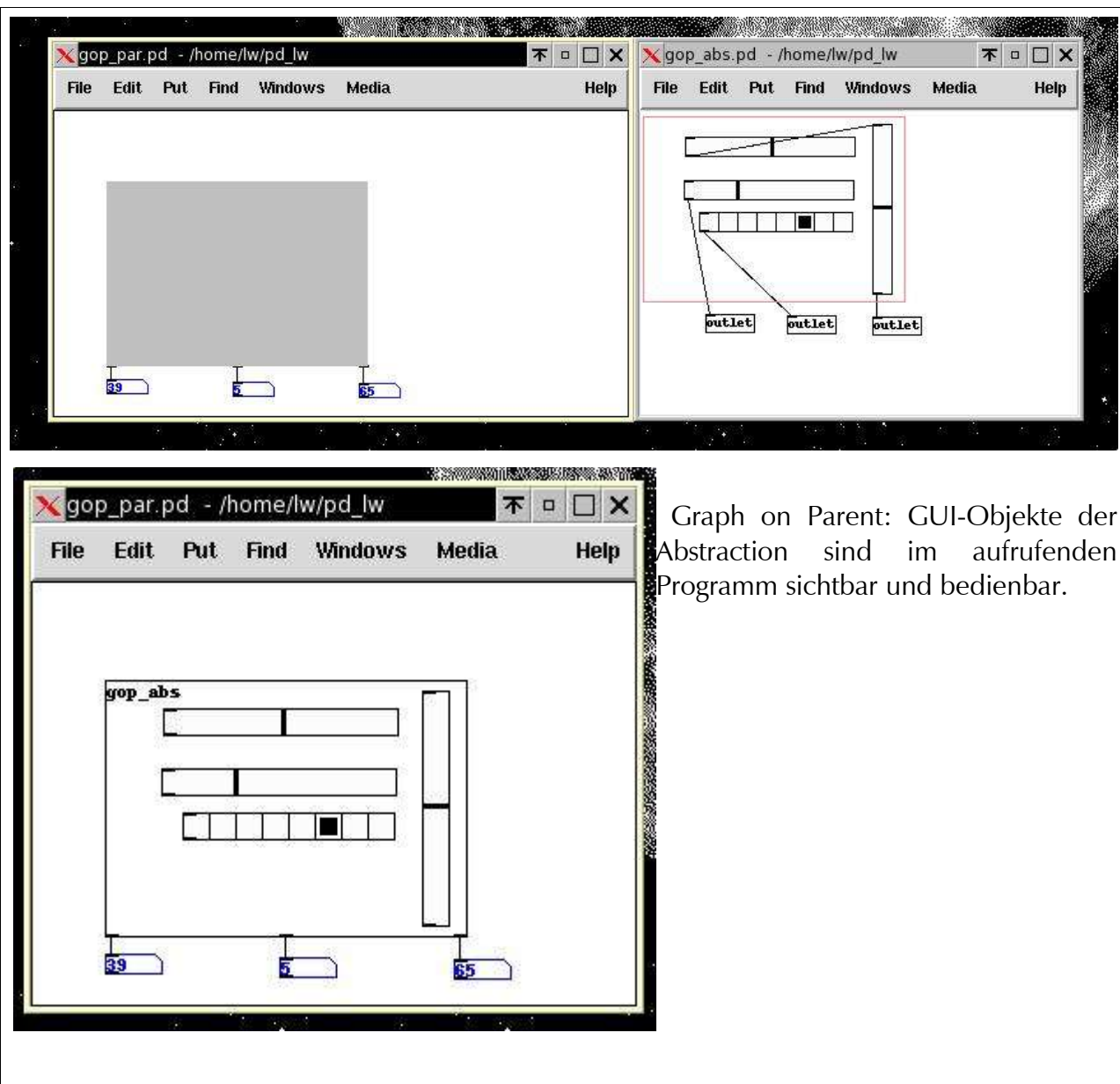
## 9. GUI-Objekte

Programmoberflächen können durch die Verwendung von Graphical-User-Interface-Objekten (ein Beitrag des IEM Institut für elektronische Musik Graz) gestaltet werden. Angeboten werden graphische Bangs, Ein/Aus-Schalter, horizontale und vertikale Slider und Radiobuttons, VU-Meter und generische Zeichenobjekte zur Verbindung mit speziellen Datenstrukturen. Über ein Kontextmenü können Farben, Größen und weitere Eigenschaften eingestellt werden.



GUI-Objekte können über eigene send/receive-Namen "drahtlos" eingesetzt werden.

GUI-Objekte in Abstractions können mittels "Graph on Parent" im Fenster des Hauptprogramms erscheinen und von dort aus bedient werden. Dazu muss im Kontextmenü des Abstraction-Hintergrunds "graph on parent" markiert werden; die Grenzen des angezeigten Bereichs erscheinen als roter Rahmen. Im Hauptprogramm sind nur GUI-Objekte und Kommentare innerhalb des Rahmens sichtbar, keine anderen Objekte oder Verbindungen.



Graph on Parent: GUI-Objekte der Abstraction sind im aufrufenden Programm sichtbar und bedienbar.