Brushed Brusselators - Principles of Construction

Franz-Josef Elmer

Januar, 2025

1 Introduction

The piece "Brushed Brusselators" is based on a dynamical system called Brusselator. The dynamical system is a system of non-linear ordinary differential equations. The solutions of these equations are time dependent functions. There values are mapped onto acoustical parameters of a set of "instruments" which generated electronical sound.

Various examples have been created which are combining three different solutions of the dynamical system (`bc1`, `bc2`, `bc3`) with three different mappings (`mapping1`, `mapping7`, `mapping8`) and two different speeds (`metro2`, `metro3`).

2 The Dynamical System

A Brusselator is a hypothetical model of chemical reactions which show periodic oscillations. The reaction schema is the following:

$$\begin{array}{rcl} \mathrm{A} & \overset{k_1}{\rightarrow} & \mathrm{X} \\ \mathrm{B}+\mathrm{X} & \overset{k_2}{\rightarrow} & \mathrm{Y}+\mathrm{C} \\ 2\mathrm{X}+\mathrm{Y} & \overset{k_3}{\rightarrow} & 3\mathrm{X} \\ \mathrm{X} & \overset{k_4}{\rightarrow} & \mathrm{D} \end{array}$$

It is assumed that the amounts of A and B are held constant. Under certain conditions of these amounts this system shows oscillations.

Brushed Brusselators is based on a **cascade of Brusselators** by assuming that the end products C and D are the A and B (or B and A) of the next Brusselator in the cascade:

$$\begin{array}{rcl} \mathrm{A_i} & \overset{k_{1,i}}{\rightarrow} & \mathrm{X_i} \\ \mathrm{B_i}+\mathrm{X_i} & \overset{k_{2,i}}{\rightarrow} & \mathrm{Y_i}+\left\{\begin{array}{ll} \mathrm{A_{i+1}} & \text{case A} \\ \mathrm{B_{i+1}} & \text{case B} \end{array}\right. \\ 2\mathrm{X_i}+\mathrm{Y_i} & \overset{k_{3,i}}{\rightarrow} & 3\mathrm{X_i} \\ \mathrm{X_i} & \overset{k_{4,i}}{\rightarrow} & +\left\{\begin{array}{ll} \mathrm{B_{i+1}} & \text{case A} \\ \mathrm{A_{i+1}} & \text{case B} \end{array}\right. \end{array}$$

The following system of ordinary differential equations models the dynamic of the amounts of the substances A, B, X and Y for each Brusselator over time:

$$\begin{aligned}
\dot{A}_i &= -k_{1,i}A_i + X_{i-1}\left\{\begin{array}{ll} k_{2,i-1}B_{i-1} & \text{case A} \\ k_{4,i-1} & \text{case B}\end{array}\right. \\
\dot{B}_i &= -k_{2,i}B_iX_i + X_{i-1}\left\{\begin{array}{ll} k_{4,i-1} & \text{case A} \\ k_{2,i-1}B_{i-1} & \text{case B}\end{array}\right. \\
\dot{X}_i &= k_{1,i}A_i - k_{2,i}B_iX_i + k_{3,i}X_i^2Y_i - k_{4,i}X_i \\
\dot{Y}_i &= k_{2,i}B_iX_i - k_{3,i}X_i^2Y_i
\end{aligned}$$

where $A_0 = B_0 = 0$.

In order to reduce the number of parameters, the following transformation has been introduced:

$$X_i = \kappa_i x_i, Y_i = \kappa_i y_i, A_i = \frac{k_{4,i}}{k_{1,i}}\kappa_i a_i, B_i = \frac{k_{4,i}}{k_{2,i}} b_i \quad \text{with} \quad \kappa_i = \sqrt{k_{4,i}/k_{3,i}}$$

This leads to the following system of ordinary differential equations:

$$\begin{aligned}
\dot{a}_i &= k_{1,i}\left(\tfrac{\gamma_{i-1}}{\gamma_i}\left\{\begin{array}{ll} b_{i-1} & \text{case A} \\ 1 & \text{case B}\end{array}\right\} x_{i-1} - a_i\right) \\
\dot{b}_i &= k_{2,i}\kappa_i\left(\tfrac{\gamma_{i-1}}{\gamma_i}\left\{\begin{array}{ll} 1 & \text{case A} \\ b_{i-1} & \text{case B}\end{array}\right\} x_{i-1} - b_ix_i\right) \\
\dot{x}_i &= k_{4,i}(a_i - b_ix_i + x_i^2y_i - x_i) \\
\dot{y}_i &= k_{4,i}(b_ix_i - x_i^2y_i)
\end{aligned}$$

where $\gamma_i = k_{4,i}\kappa_i$. We absorb κ_i into $k_{2,i}$, that is we replace $k_{2,i}\kappa_i$ by $k_{2,i}$. In addition we introduce $T_i = 1/k_{4,i}$. Thus, the final system for a cascade of N Brusselator reads:

$$\begin{aligned}
\dot{a}_i &= k_{1,i}\left(\tfrac{\gamma_{i-1}}{\gamma_i}\left\{\begin{array}{ll} b_{i-1} & \text{case A} \\ 1 & \text{case B}\end{array}\right\} x_{i-1} - a_i\right) \\
\dot{b}_i &= k_{2,i}\left(\tfrac{\gamma_{i-1}}{\gamma_i}\left\{\begin{array}{ll} 1 & \text{case A} \\ b_{i-1} & \text{case B}\end{array}\right\} x_{i-1} - b_ix_i\right) \\
\dot{x}_i &= (a_i - b_ix_i + x_i^2y_i - x_i)/T_i \\
\dot{y}_i &= (b_ix_i - x_i^2y_i)/T_i
\end{aligned}$$

with $a_0 = b_0 = 0$. The parameters are $k_{1,i}$, $k_{2,i}$, γ_i, T_i and the case (A or B), for $i = 1, 2, \cdot, N$.

This system of ordinary differential equation is integrated numerically by a python script using the fourth order Runge-Kutta algorithm. The result is stored in a text file.

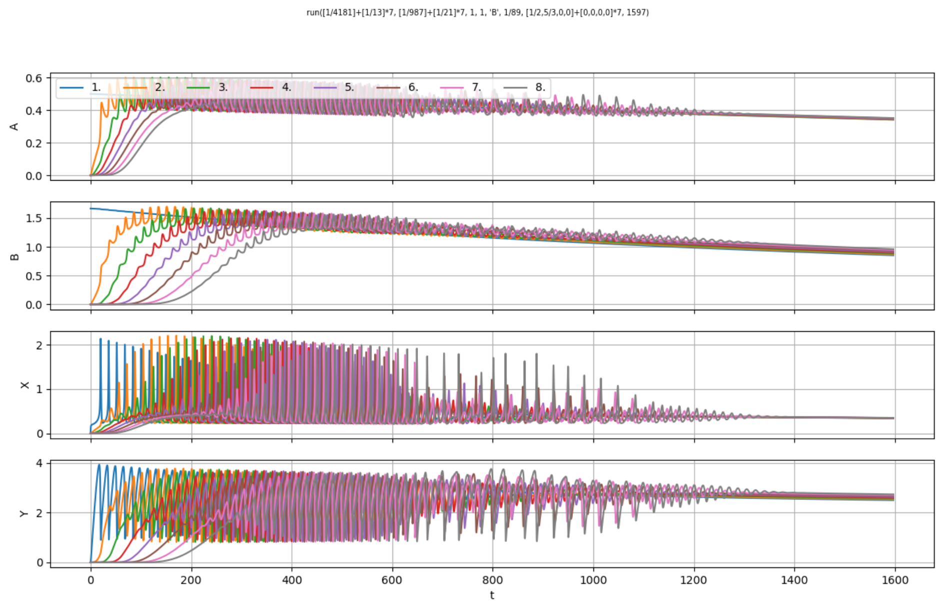
The three different solutions vary in parameters and intial conditions but have the following in common:

$$\begin{aligned}
N &= 8, \\
\text{case} &= \text{B}, \\
a_i(0) &= 0, \quad \text{for all } i > 1, \\
b_i(0) &= 0, \quad \text{for all } i > 1, \\
x_i(0) &= 0, \quad \text{for all } i,
\end{aligned}$$

$$\begin{aligned} y_i(0) &= 0, \quad \text{for all } i, \\ \Delta T &= 1/89, \\ \text{Total integration time} &= 1597. \end{aligned}$$

`bc1`:

$$\begin{aligned} \kappa_{1,1} &= 1/4181, \\ \kappa_{1,i} &= 1/13, \quad \text{for } i > 1, \\ \kappa_{2,1} &= 1/987, \\ \kappa_{2,i} &= 1/21, \quad \text{for } i > 1, \\ \gamma_i &= 1, \quad \text{for all } i, \\ T_i &= 1, \quad \text{for all } i, \\ a_1(0) &= 1/2, \\ b_1(0) &= 5/3. \end{aligned}$$



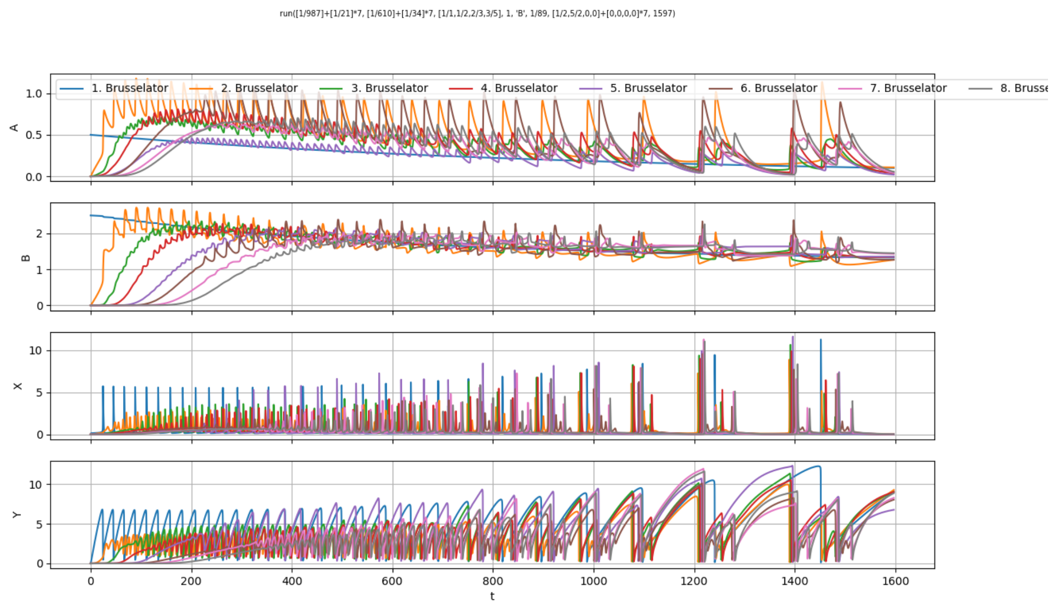
```
run([1/4181]+[1/13]*7, [1/987]+[1/21]*7, 1, 1, 'B', 1/89, [1/2,5/3,0,0]+[0,0,0,0]*7, 1597)
```

Figure 1: Plots of a_i, b_i, x_i, and y_i for `bc1`.

`bc2`:

$$\begin{aligned} \kappa_{1,1} &= 1/987, \\ \kappa_{1,i} &= 1/21, \quad \text{for } i > 1, \\ \kappa_{2,1} &= 1/610, \\ \kappa_{2,i} &= 1/34, \quad \text{for } i > 1, \\ \gamma_1 &= 1, \\ \gamma_2 &= 1/2, \end{aligned}$$

$$
\begin{aligned}
\gamma_3 &= 2/3,\\
\gamma_4 &= 3/5,\\
\gamma_5 &= 1,\\
\gamma_6 &= 1/2,\\
\gamma_7 &= 2/3,\\
\gamma_8 &= 3/5,\\
T_i &= 1, \quad \text{for all } i,\\
a_1(0) &= 1/2,\\
b_1(0) &= 5/2.
\end{aligned}
$$



```
run([1/987]+[1/21]*7, [1/610]+[1/34]*7, [1/1,1/2,2/3,3/5], 1, 'B', 1/89, [1/2,5/2,0,0]+[0,0,0,0]*7, 1597)
```

Figure 2: Plots of a_i, b_i, x_i, and y_i for `bc2`.

`bc3`**:**

$$
\begin{aligned}
\kappa_{1,1} &= 0,\\
\kappa_{1,i} &= 1/34, \quad \text{for } i > 1,\\
\kappa_{2,1} &= 1/377,\\
\kappa_{2,i} &= 1/55, \quad \text{for } i > 1,\\
\gamma_i &= 1, \quad \text{for all } i,\\
T_1 &= 1,\\
T_2 &= 1,\\
T_3 &= 2,\\
T_4 &= 3,\\
T_5 &= 5,
\end{aligned}
$$

$$
\begin{aligned}
T_6 &= 8, \\
T_7 &= 13, \\
T_8 &= 21, \\
a_1(0) &= 1/2, \\
b_1(0) &= 5/2.
\end{aligned}
$$

```
run([0]+[1/34]*7, [1/377]+[1/55]*7, 1, [1,1,2,3,5,8,13,21], 'B', 1/89, [1/2,5/2,0,0]+[0,0,0,0]*7, 1597)
```

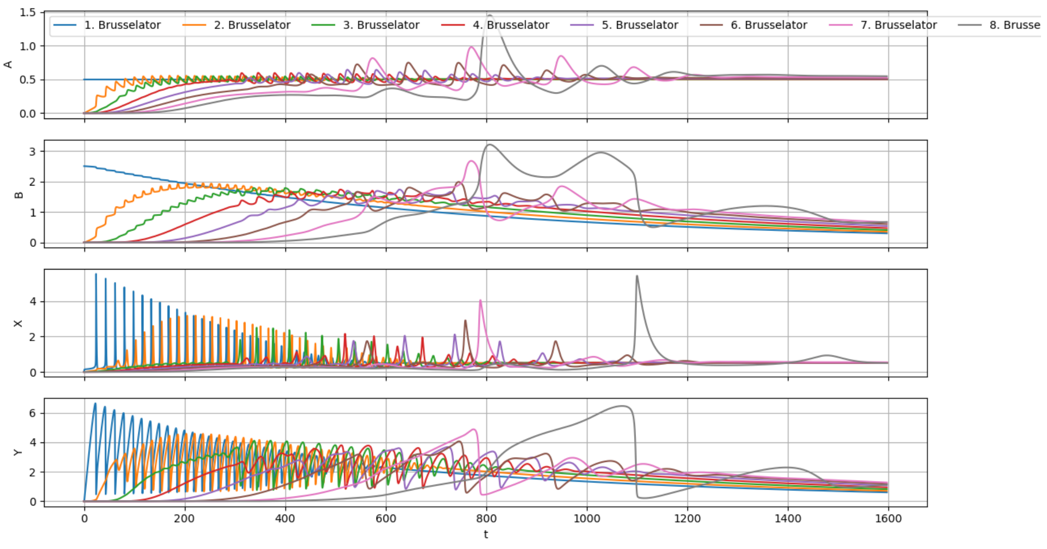


Figure 3: Plots of a_i, b_i, x_i, and y_i for `bc3`.

Note: the total length of an example in seconds is given by *Total integration time*$/\Delta t \cdot$ *metro time in msec*. This means for the two different metro values:

`metro2`: $1597 \cdot 89 \cdot 2/1000 = 284$ sec $= 4$ min 44 sec

`metro3`: $1597 \cdot 89 \cdot 3/1000 = 426$ sec $= 7$ min 6 sec.

3 Mapping

A Max/MSP patcher reads the result of the integrated dynamical system and uses mappings to create the actual sound. The mapping is specified in a text file.

The mapping file contains the mapping for an arbitrary number of instruments. Each instrument is assigned to one of the brusselators. That is, only the values of $a_i(t)$, $b_i(t)$, $x_i(t)$, and $y_i(t)$ of brusselator i are used to specify the sound.

There are three different instrument types, which are defined in Max/MSP subpatches. Each instrument type has specific parameters. Functions are defined to map the above mentioned values of the brusselator i onto the parameters of the instrument type.

The syntax of a function in the mapping file is the following:

```
<instrument number>.<parameter name> = <expression>
```

where `<instrument number>` is the instrument for which the mapping is specified, `<parameter name>` is the name of a parameter of the specified instrument, and `<expression>` is an expression which determines the value of the parameter based on the values of a, b, x, and y.

The set of possible parameter names are usually different for the different instrument types. But the following parameters are common for all instrument types:

`type`: The number of the instrument type. Possible values are 1, 2 or 3.

`brusselator`: The number of brusselator. If unspecified, `brusselator = <instrument number>`.

`stereo`: Defines where the sound source is positioned. 0 means the sound comes only from the left loudspeaker. 1 means the opposite.

The `<expression>` is any valid Javascript expression. The following functions can be used:

`a(t)`: The value of $a_{\texttt{brusselator}}$ at the current time plus `t`. `t` is a relative time index, e.g. $\texttt{t} = -3$ means the data three time steps before the current time should be used. If `t` isn't specified the current time will be used.

`b(t)`: The value of $b_{\texttt{brusselator}}$ at the current time plus `t`.

`x(t)`: The value of $x_{\texttt{brusselator}}$ at the current time plus `t`.

`y(t)`: The value of $y_{\texttt{brusselator}}$ at the current time plus `t`.

`maximum(n, m)`: Maximum function, i.e. `Math.max(n,m)`.

`minimum(n, m)`: Minimum function, i.e. `Math.min(n,m)`.

`abs(n)`: Absolute function, i.e. `Math.abs(n)`.

`round(n)`: Rounding function, i.e. `Math.round(n)`.

`sqrt(n)`: Square root function, i.e. `sqrt(n)`.

`pow(n, m)`: Power function $\texttt{n}^{\texttt{m}}$, i.e. `pow(n,m)`.

`step(z, dz)`: Smoothed round function where steps to the next integer value are smoothed out in accordance to `dz`. That is, $\texttt{step}(\texttt{z}, 0) = \texttt{round}(\texttt{z})$.

`smoothMax0(z, dz)`: Smoothed maximum function `maximum(z)`, where the transition at `z` = 0 is smoothed out in accordance to `dz`. That is, smoothMax0(z, 0) = maximum(z, 0).

A type 1 instrument is a sinusoidal oscillator which phase modulation by another sinusoidal oscillator. Parameters:

`freq`: Frequency of the primary oscillator.

`amp`: Amplitude of the primary oscillator. Default value: 0.2.

`mod-freq`: Modulation frequency.

`mod-amp`: Modulation amplitude.

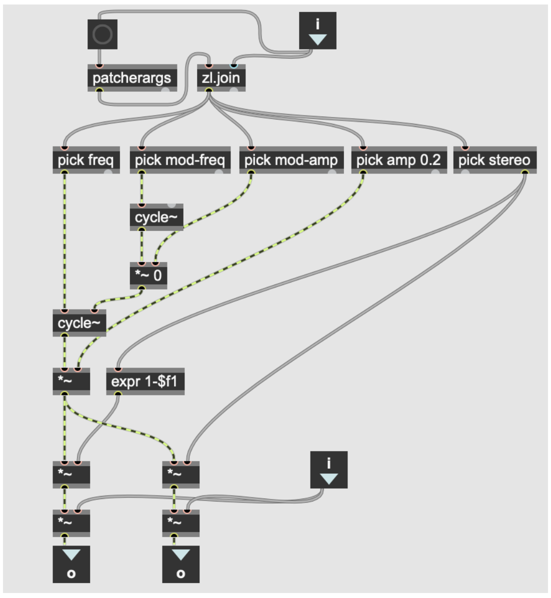


Figure 4: Max/MSP subpatcher for instruments of type 1

A type 2 instrument produces a decaying tone of a certain pitch in regular time intervals. The tone comes from a frequency modulated triangular oscillator. Parameters:

`metro`: Length of the time interval in msec. Default value: 2000.

`freq`: Frequency of the triangular oscillator.

`mod-freq`: Modulation frequency.

`freq-amp-end`: End value of a linear increase of the modulation amplitude which starts at 0 and ends after `freq-amp-end` msec. Default value: 10.

`freq-amp-time`: Time in msec when the increase of the modulation amplitude is stopped. Default value: 400.

`dc-start`: Start value of the so-called duty cycle which determines the position of the maximum of the triangular oscillation relative to the minima. Possible values are between 0 and 1. Default value: 0.5.

`dc-end`: The duty cycle changes linearly in `dc-time` msec from `dc-start` to `dc-end`. Default value: 0.

`dc-time`: Time in msec when the change of the duty cycle is stopped. Default value: 700.

`amp`: Maximum amplitude of the triangular oscillation after `attack` msec. Default value: 0.2.

`attack`: Time in msec after the maximum of the amplitude has been reached (starting at 0). Default value: 10.

`decay`: Time in msec after which the amplitude decays to zero. Default value 1000.

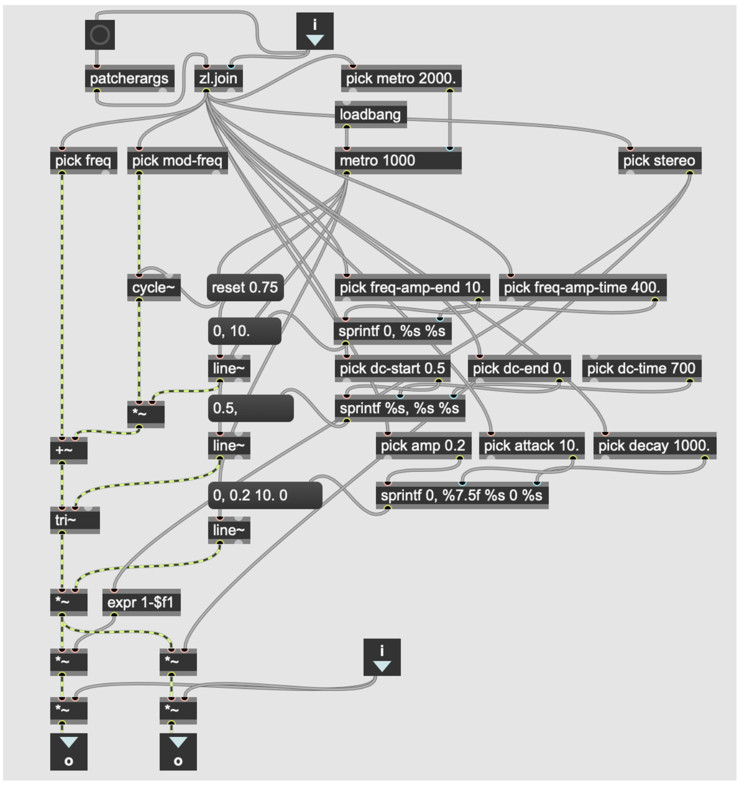


Figure 5: Max/MSP subpatcher for instruments of type 2

A type 3 instrument is a frequency modulated triangular oscillator. The frequency is modulated by a sinusoidal oscillator. Parameters:

`freq`: Frequency of the triangular oscillator.

`amp`: Amplitude of the triangular oscillator. Default value: 0.2.

`mod-freq`: Modulation frequency.

`mod-amp`: Modulation amplitude.

`duty-cycle`: Defines the position of the maximum in a period of two minima. Values between 0 and 1 are valid. Value 0.5 means that the maximum is in the middle between the two minima. Default value: 0.5.

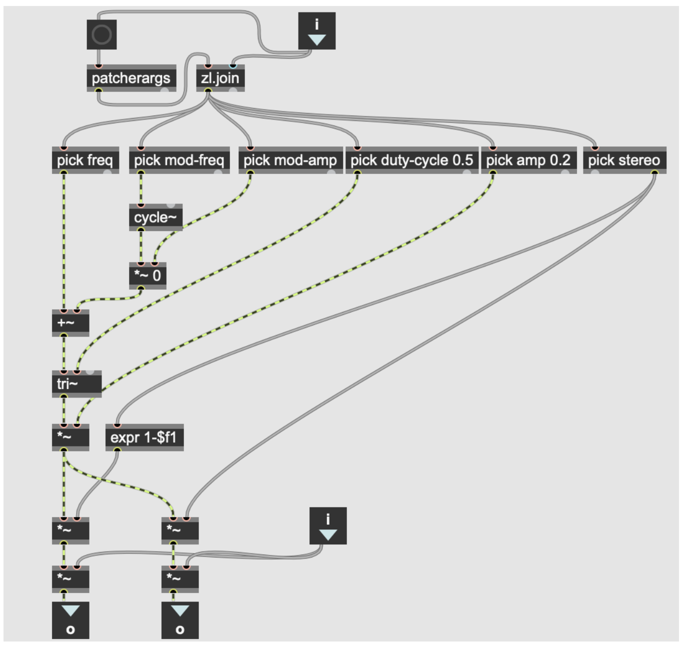


Figure 6: Max/MSP subpatcher for instruments of type 3

mapping1:

```
1.type = instrument3
1.freq = 89 * pow(2, step(2 * y(), 1/13) / 8)
1.mod-freq = 2 * y()
1.mod-amp = 8 * b()
1.duty-cycle = 1/3
1.amp = smoothMax0((x() - a())/5, 1/377)
1.stereo = 1/7
#
2.type = instrument1
2.freq = 610 /  pow(2, step(2 * y(), 1/13) / 8)
2.mod-freq = 2 * y()
2.mod-amp = 8 * b()
2.amp = smoothMax0((x() - a())/21, 1/377)
2.stereo = 6/7
#
3.type = instrument3
3.freq = 610 / pow(2, step(2 * y(), 1/13) / 8)
3.mod-freq = 3 * y()
3.mod-amp = 8 * b()
3.duty-cycle = 1/2
3.amp = smoothMax0((x() - a())/8, 1/377)
3.stereo = 5/7
```

```
#
4.type = instrument1
4.freq = 144 * pow(2, step(2 * y(), 1/13) / 8)
4.mod-freq = 3 * y()
4.mod-amp = 8 * b()
4.amp = smoothMax0((x() - a())/21, 1/377)
4.stereo = 2/7
#
5.type = instrument3
5.freq = 987 / pow(2, step(2 * y(), 1/13) / 8)
5.mod-freq = 5 * y()
5.mod-amp = 8 * b()
5.duty-cycle = 1/5
5.amp = smoothMax0((x() - a())/21, 1/377)
5.stereo = 3/7
#
6.type = instrument1
6.freq = 233 *  pow(2, step(2 * y(), 1/13) / 8)
6.mod-freq = 5 * y()
6.mod-amp = 8 * b()
6.amp = smoothMax0((x() - a())/21, 1/377)
6.stereo = 4/7
#
7.type = instrument2
7.metro = 1597 * maximum(b(), 1/8)
7.freq = 233 * pow(2, step(2 * y(), 1/13) / 8)
7.mod-freq = 2
7.freq-amp-end = 13 * x()
7.freq-amp-time = 233 * a()
7.dc-end = 2/3
7.amp = abs(x() - a()) / 5
7.decay =  987 * maximum(b(), 1/13)
7.stereo = 0
#
8.type = instrument2
8.metro = 987 * maximum(b(), 1/8)
8.freq = 987 / pow(2, step(y(), 1/13) / 5)
8.mod-freq = 3
8.freq-amp-end = 8 * x()
8.freq-amp-time = 144 * a()
8.dc-end = 1/2
8.amp = abs(x() - a()) / 5
8.decay = 610 * maximum(b(), 1/13)
8.stereo = 1
```

mapping7:

```
1.type = instrument3
1.freq = 987 / pow(2, step(21 * (b() - a()), 0.1) / 13)
1.mod-freq = 21 * y()
1.mod-amp = 21 * b()
1.duty-cycle = 0.5 + 3 * (0.5 - a())
1.amp = maximum(x() - a(), 1/89) / 3
1.stereo = 0/7
#
2.type = instrument3
2.freq = 987 / pow(2, step(21 * (b() - a()), 0.1) / 13)
2.mod-freq = 21 * y()
2.mod-amp = 21 * b()
2.duty-cycle = 0.5 + 3 * (0.5 - a())
2.amp = maximum(x() - a(), 1/89) / 3
2.stereo = 1/7
#
3.type = instrument3
```

```
3.freq = 987 / pow(2, step(21 * (b() - a()), 0.1) / 13)
3.mod-freq = 21 * y()
3.mod-amp = 21 * b()
3.duty-cycle = 0.5 + 3 * (0.5 - a())
3.amp = maximum(x() - a(), 1/89) / 3
3.stereo = 2/7
#
4.type = instrument3
4.freq = 987 / pow(2, step(21 * (b() - a()), 0.1) / 13)
4.mod-freq = 21 * y()
4.mod-amp = 21 * b()
4.duty-cycle = 0.5 + 3 * (0.5 - a())
4.amp = maximum(x() - a(), 1/89) / 3
4.stereo = 3/7
#
5.type = instrument3
5.freq = 987 / pow(2, step(21 * (b() - a()), 0.1) / 13)
5.mod-freq = 21 * y()
5.mod-amp = 21 * b()
5.duty-cycle = 0.5 + 3 * (0.5 - a())
5.amp = maximum(x() - a(), 1/89) / 3
5.stereo = 4/7
#
6.type = instrument3
6.freq = 987 / pow(2, step(21 * (b() - a()), 0.1) / 13)
6.mod-freq = 21 * y()
6.mod-amp = 21 * b()
6.duty-cycle = 0.5 + 3 * (0.5 - a())
6.amp = maximum(x() - a(), 1/89) / 3
6.stereo = 5/7
#
7.type = instrument3
7.freq = 987 / pow(2, step(21 * (b() - a()), 0.1) / 13)
7.mod-freq = 21 * y()
7.mod-amp = 21 * b()
7.duty-cycle = 0.5 + 3 * (0.5 - a())
7.amp = maximum(x() - a(), 1/89) / 3
7.stereo = 6/7
#
8.type = instrument3
8.freq = 987 / pow(2, step(21 * (b() - a()), 0.1) / 13)
8.mod-freq = 21 * y()
8.mod-amp = 21 * b()
8.duty-cycle = 0.5 + 3 * (0.5 - a())
8.amp = maximum(x() - a(), 1/89) / 3
8.stereo = 7/7
```

mapping8:

```
1.type = instrument1
1.freq = 233 * pow(2, 5 * y() / 13)
1.mod-freq = b() + 1/2
1.mod-amp = 1 / (a() + 1/21)
1.amp = maximum(x(55) - a(55), 1/89) / 8
1.stereo = 0/7
#
2.type = instrument1
2.freq = 233 * pow(2, 5 * y() / 13)
2.mod-freq = b() + 1/2
2.mod-amp = 1 / (a() + 1/21)
2.amp = maximum(x(55) - a(55), 1/89) / 8
2.stereo = 1/7
#
3.type = instrument1
```

```
3.freq = 233 * pow(2, 5 * y() / 13)
3.mod-freq = b() + 1/2
3.mod-amp = 1 / (a() + 1/21)
3.amp = maximum(x(55) - a(55), 1/89) / 8
3.stereo = 2/7
#
4.type = instrument1
4.freq = 233 * pow(2, 5 * y() / 13)
4.mod-freq = b() + 1/2
4.mod-amp = 1 / (a() + 1/21)
4.amp = maximum(x(55) - a(55), 1/89) / 8
4.stereo = 3/7
#
5.type = instrument1
5.freq = 233 * pow(2, 5 * y() / 13)
5.mod-freq = b() + 1/2
5.mod-amp = 1 / (a() + 1/21)
5.amp = maximum(x(55) - a(55), 1/89) / 8
5.stereo = 4/7
#
6.type = instrument1
6.freq = 233 * pow(2, 5 * y() / 13)
6.mod-freq = b() + 1/2
6.mod-amp = 1 / (a() + 1/21)
6.amp = maximum(x(55) - a(55), 1/89) / 8
6.stereo = 5/7
#
7.type = instrument1
7.freq = 233 * pow(2, 5 * y() / 13)
7.mod-freq = b() + 1/2
7.mod-amp = 1 / (a() + 1/21)
7.amp = maximum(x(55) - a(55), 1/89) / 8
7.stereo = 6/7
#
8.type = instrument1
8.freq = 233 * pow(2, 5 * y() / 13)
8.mod-freq = b() + 1/2
8.mod-amp = 1 / (a() + 1/21)
8.amp = maximum(x(55) - a(55), 1/89) / 8
8.stereo = 7/7
```