$\mathbb{R}^2$m user's manual

for version 0.1.0

Fritz Menzer

fritz.menzer@epfl.ch

31. 3. 2001

Contents

1 What is $\mathbb{R}^2$m? **2**
1.1 What is the $\mathbb{R}^2$m program? . . . 2

2 Installation of the $\mathbb{R}^2$m program **3**
2.1 Requirements . . . 3
2.2 Download and installation . . . 3
2.3 Deinstallation . . . 3

3 User interface of the $\mathbb{R}^2$m program **4**
3.1 Overview . . . 4
3.2 Mapping function . . . 5
3.2.1 Editing the mapping function . . . 5
3.2.2 Defining the displayed area . . . 5
3.3 Path generation . . . 6
3.3.1 Path definition and frequency . . . 6
3.3.2 Amplitude and Center . . . 6
3.3.3 Envelope . . . 7
3.3.4 Starting point . . . 7
3.4 Voices, MIDI channel and sound storage . . . 8
3.5 MIDI connection . . . 8

4 $\mathbb{R}^2$m sound programming techniques **8**

A File formats **9**
A.1 r2m.dtd . . . 9
A.2 Mapping function file example . . . 10
A.3 Sound file example . . . 10

1 What is $\mathbb{R}^2$m?

$\mathbb{R}^2$m stands for $\mathbb{R}^2$ mapping and is a sound synthesis method that allows to produce a wide range of dynamic sounds.

The principle is easy to understand. We produce two independent signals $x(t)$ and $y(t)$. Each signal has variable amplitude and offset. $(x(t), y(t))$ can be seen as a path in $\mathbb{R}^2$ (the two dimensional plane). To produce a sound we use a mapping function $f(x, y)$ that maps the path $(x(t), y(t))$ on the interval $[-1, +1]$ so that $s(t) = f(x(t), y(t))$ is a sound signal (which can be sent to a loudspeaker for example).

A possible method to generate the x and y signals is described in following formula:

$$\begin{aligned} x(t) &= A_x(t)cos(\omega_x t) + C_x(t) \\ y(t) &= A_y(t)sin(\omega_y t) + C_y(t) \end{aligned} \tag{1}$$

In equation 1 $C_x(t)$ and $C_y(t)$ are offset (or center) control signals and $A_x(t)$ and $A_y(t)$ are amplitude control signals.

The mapping function f can be any function

$$\begin{aligned} f : \mathbb{R}^2 &\to \mathbb{R} \\ (x, y) &\mapsto f(x, y) \end{aligned} \tag{2}$$

such that

$$\forall (x, y) \in \mathbb{R}^2, \ f(x, y) \in [-1, +1] \tag{3}$$

An easy way to achieve this is to define

$$f(x, y) = \min\left\{\max\left\{g(x, y), -1\right\}, +1\right\} \tag{4}$$

where $g(x, y)$ has no constraints on its values.

1.1 What is the $\mathbb{R}^2$m program?

The $\mathbb{R}^2$m program is a program that implements the $\mathbb{R}^2$m synthesis method. It runs on BeOS and is available as a beta[1] version at `http://www.xsmusic.ch` for free.

[1] The beta version of the $\mathbb{R}^2$m program may only be used until the 1^{st} of september 2001

2 Installation of the $\mathbb{R}^2$m program

2.1 Requirements

To be able to install the $\mathbb{R}^2$m program on your computer you have to have BeOS 5.0 or higher installed on your computer. Furthermore you need to have the development tools installed, notably gcc, the GNU C Compiler which is available at `http://www.bebits.com`. If you have the "Pro" edition of BeOS, you don't have to worry - gcc is already included.

As this is a very early version of the $\mathbb{R}^2$m program it is absolutely not optimized, so don't be surprised if you get only about 10 voices[2] out of a Pentium III 700Mhz. But there is still a lot of potential to optimize the synthesis engine.

2.2 Download and installation

The first thing you have to do is download `r2m.zip` at `http://www.xsmusic.ch`. Then you should expand the archive on your BeOS partition, for example in the home directory `/boot/home`. This will create a new directory called `R2m` that contains all the files necessary to run the $\mathbb{R}^2$m program (except gcc).

When you start $\mathbb{R}^2$m for the first time it will ask you to read the licensing agreement. In order to use the program you have to accept the licensing agreement.

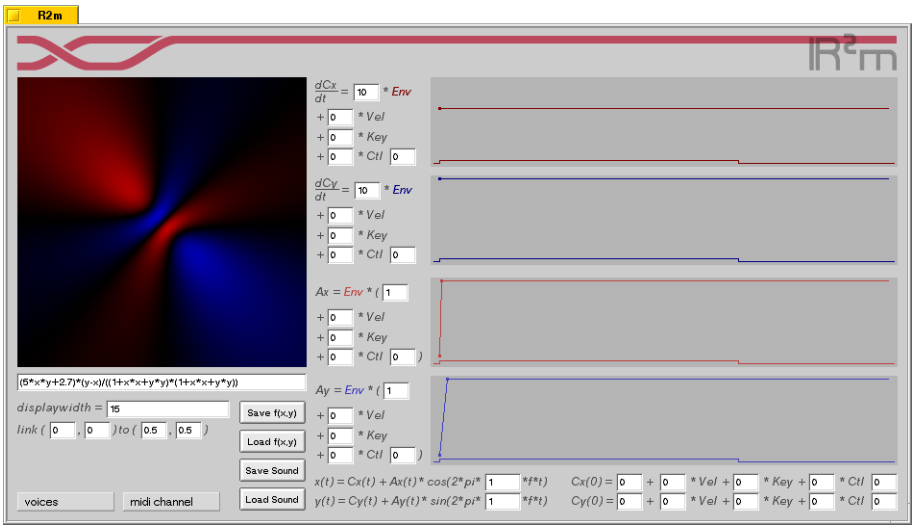
2.3 Deinstallation

If for some reason you want to deinstall the $\mathbb{R}^2$m program, the only thing you need to do is to delete the `R2m` directory. That's the ease of use of BeOS... Just be sure to save the sounds and functions in the `sounds` and `functions` directories if you want to keep them.

[2]The number of voices depends on the complexity of the sound, especially of the mapping function and on the number of envelope points

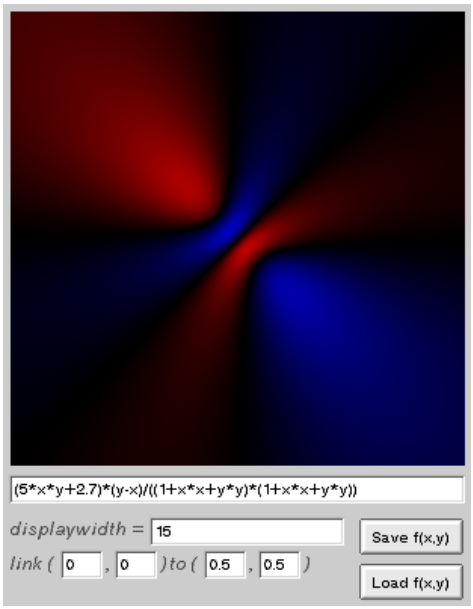
3 User interface of the $\mathbb{R}^2$m program

3.1 Overview



The $\mathbb{R}^2$m user interface is generally divided in two parts: To the left you can define the mapping function and the way it is displayed while everything on the right hand side of the function display is needed to define the path $(x(t), y(t))$.

3.2 Mapping function



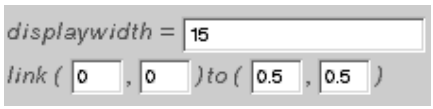
The mapping function is displayed using a colormap. If a point is red, the function takes the value -1 at this point. Blue stands for $+1$ and black for 0. Intermediate colors (for example dark blue) stand for intermediate values.

3.2.1 Editing the mapping function

To change the mapping function, just type a new function in the text field below the function display and press enter. It may take a few seconds until the function is comiled.

To load and save the mapping function you can use the buttons labeled Load f(x,y) and Save f(x,y).

3.2.2 Defining the displayed area



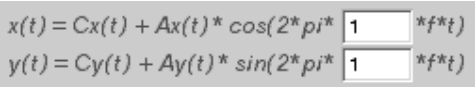
You can tell $\mathbb{R}^2$m what area you want to have displayed by indicating the width of the display (in the function's coordinates) and by specifying which point of the function should be linked to which point on the display. The point of the function is specified in the function's coordinates and the point on the display in a coordinate system which has its origin in the bottom left corner and in which the display's width is 1.

For example link $(0, 0)$ to $(0, 1)$ would mean that you want to link the origin of the function $(0, 0)$ to the top left corner $(0, 1)$ of the display.

This system has the advantage that you can link a point of interest of the function to a certain point on the display and change only the displaywidth parameter to zoom in and out.

3.3 Path generation

3.3.1 Path definition and frequency



This part of the user interface shows the formula used to calculate $x(t)$ and $y(t)$ and lets you specify the frequencies of the sine wave oscillators. The f to the right of the editable text fields stands for the base frequency of the sound, so if you want to have both oscillators fixed to the base frequency, you just type 1 in both fields.

3.3.2 Amplitude and Center

The amplitude control signals are generated by envelopes ($Env(t)$). These envelopes produce values between 0 and +1:

$$Env(t) \in [0, +1]$$

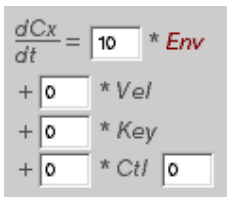
Each amplitude control signal is derived from the corresponding envelope according to the following formula (where $A(t)$ may stand for $A_x(t)$ or $A_y(t)$):

$$A(t) = Env(t)\,(p_1 + p_2 Vel + p_3 Key + p_4 Ctl[p_5](t)) \tag{5}$$

In (5) Vel and Key are values that are associated to each note played and represent the key velocity and the key number. They are normalized to $[0, +1]$, so that MIDI velocity 0 corresponds to $Vel = 0$, 127 to $Vel = 1$, MIDI note 0 (C_3) to $Key = 0$ and note 127 (g^6) to $Key = 1$. $Ctl[p_5](t)$ is the MIDI controller p_5.

Each time a control change message whose controller number is equal to p_5 is received $Ctl[p_5]$ changes its value. Because there are only 128 valid controller numbers, p_5 must be an integer in $\{0, 2, ..., 127\}$.

The parameters $p_1, ..., p_4$ are floating point numbers, i.e. approximations of real numbers.



The center control signals are generated the same way as the amplitude control signals, except for one big difference: Instead of defining the center control signal

directly, you define its time derivative, as to say the *velocity* of the center. The real center control signals are calculated by integration:

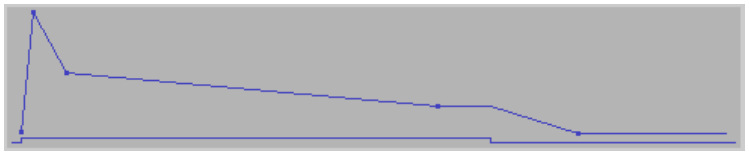
$$C(t) = \int_0^t \frac{dC(t)}{dt} dt + C(0) \tag{6}$$

This makes it possible - and very easy - to define for example a path where the center moves with constant velocity on a straight line after the key was released. Just set a single flat segment in the release part of each envelope associated with the $\frac{dC_x(t)}{dt}$ and $\frac{dC_y(t)}{dt}$ signals (i.e. the speed in the x and y directions).

$$\frac{dC(t)}{dt} = p_1 Env(t) + p_2 Vel + p_3 Key + p_4 Ctl[p_5](t) \tag{7}$$

In (7) $Env(t)$ is in $[-1, +1]$ while the other parameters and signals are defined as for equation (5). $C(t)$ may stand for $C_x(t)$ or $C_y(t)$. Notice also the structural difference between equations (5) and (7).

3.3.3 Envelope



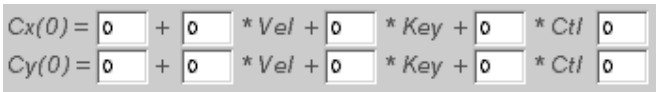
$\mathbb{R}^2$m's envelopes are linear and can have an unlimited number of segments. Be aware that a high number of segments means also a high CPU load, and therefore less voices. This will hopefully be fixed in a future version.

At the bottom of each envelope there is a line indicating an imaginary "key-pressed" signal. "High" means the key is pressed, "low" means it is not. This divides the envelope in two parts: When the key is pressed, the envelelope generates the signal defined in the "high" part and when the key is released it starts immediately with the "low" part.

To add a segment to the envelope, just left-click in the envelope and a new cornerpoint is generated. To remove a cornerpoint, right-click on it. To move a point, drag it with the left mouse button.

To get a better resolution in the envelope you can stretch the $\mathbb{R}^2$m window or edit the sound file directly (see appendix A).

3.3.4 Starting point



As you may have noticed in equation (6) there is a term $C(0)$. That means that for each center control signal $C(t)$ in addition to the $\frac{dC(t)}{dt}$ signal we also need a value that defines $C(t)$ for $t = 0$ (the instant when the key is pressed). If you know the velocity of a point at any time, you need to know where the point was at a certain instant to be able to say where the point is at any time.

So $C_x(0)$ and $C_y(0)$ define the starting point of the center control signal. These two values may depend on the key number, the velocity and a controller in a way similar to equations (5) and (7):

$$C(0) = p_1 + p_2 Vel + p_3 Key + p_4 Ctl[p_5](0) \tag{8}$$

where $C(0)$ may stand for $C_x(0)$ or $C_y(0)$. The parameters $p1, ..., p5$ are defined as in equation (7)

Remark: $(C_x(0), C_y(0))$ is not necessarily the starting point of the *path*, due to equation (1). The starting point of the path is in fact

$$\begin{aligned} x(0) &= A_x(0) + C_x(0) \\ y(0) &= \quad C_y(0) \end{aligned} \tag{9}$$

3.4 Voices, MIDI channel and sound storage

You can define how many voices $\mathbb{R}^2$m should produce simultaneously and from which MIDI channel it should get the data. If the CPU power is not enough to produce the desired number of voices, the synthesis engine will automatically turn off voices[3]. The default value for the maximum number of voices and for the MIDI channel is 1.

To load and save sounds you can use the buttons labeled Load Sound and Save Sound.

3.5 MIDI connection

To connect $\mathbb{R}^2$m's MIDI input to some MIDI producer node (such as a MIDI interface) you have to use the program PatchBay (at least for now) which is released by Be Inc. under Be Sample Code License and comes with the $\mathbb{R}^2$m program.

4 $\mathbb{R}^2$m sound programming techniques

Programming sounds with $\mathbb{R}^2$m is a nearly unlimited topic (because there is an infinity of possible mapping functions) that merits being treated in a seperate document. Please refer to http://www.xsmusic.ch/articles/r2mspt.pdf.

[3]The synthesis engine starts to turn off voices as soon as the calculation of the voices takes up more than 80% of the CPU time

A File formats

$\mathbb{R}^2$m saves its mapping functions and sounds in XML format. This means that with a simple text editor you can modify these files. This is especially useful for sound files, because you can edit the envelopes at an arbitrary precision.

A.1 r2m.dtd

```
<!ELEMENT r2mfunction          (info, expression, display)>
<!ELEMENT r2msound             (info, r2mfunction, dcxdt, dcydt, ax,
                                ay, cx0, cy0, fx, fy)>

<!ELEMENT info                 (r2mversion, author)>
<!ELEMENT expression           (#PCDATA)>
<!ELEMENT display              (width, linkfunction, linkscreenrel)>

<!ELEMENT r2mversion           (#PCDATA)>
<!ELEMENT author               (#PCDATA)>

<!ELEMENT width                (#PCDATA)>
<!ELEMENT linkfunction         (x, y)>
<!ELEMENT linkscreenrel        (x, y)>

<!ELEMENT x                    (#PCDATA)>
<!ELEMENT y                    (#PCDATA)>

<!ELEMENT dcxdt                (env, vel, key, ctl, ctlnum, envdata)>
<!ELEMENT dcydt                (env, vel, key, ctl, ctlnum, envdata)>
<!ELEMENT ax                   (env, vel, key, ctl, ctlnum, envdata)>
<!ELEMENT ay                   (env, vel, key, ctl, ctlnum, envdata)>
<!ELEMENT cx0                  (one, vel, key, ctl, ctlnum)>
<!ELEMENT cy0                  (one, vel, key, ctl, ctlnum)>
<!ELEMENT fx                   (#PCDATA)>
<!ELEMENT fy                   (#PCDATA)>

<!ELEMENT env                  (#PCDATA)>
<!ELEMENT one                  (#PCDATA)>
<!ELEMENT vel                  (#PCDATA)>
<!ELEMENT key                  (#PCDATA)>
<!ELEMENT ctl                  (#PCDATA)>
<!ELEMENT ctlnum               (#PCDATA)>

<!ELEMENT envdata              (envpoint+)>

<!ELEMENT envpoint             EMPTY>

<!ATTLIST envpoint  t   CDATA          #REQUIRED
                    val CDATA          #REQUIRED
                    fku (true | false) #REQUIRED>
```

A.2 Mapping function file example

```
<?xml version="1.0"?>
<!DOCTYPE r2mfunction SYSTEM "http://www.xsmusic.ch/dtd/r2m.dtd">

<r2mfunction>
  <info>
    <r2mversion>0.1beta</r2mversion>
    <author>Fritz Menzer</author>
  </info>
  <expression>(5*x*y+2.7)*(y-x)/((1+x*x+y*y)*(1+x*x+y*y))</expression>
  <display>
    <width>10</width>
    <linkfunction>
      <x>0</x>
      <y>0</y>
    </linkfunction>
    <linkscreenrel>
      <x>0.5</x>
      <y>0.5</y>
    </linkscreenrel>
  </display>
</r2mfunction>
```

A.3 Sound file example

```
<?xml version="1.0"?>
<!DOCTYPE r2msound SYSTEM "http://www.xsmusic.ch/dtd/r2m.dtd">

<r2msound>
  <info>
    <r2mversion>0.1beta</r2mversion>
    <author>Fritz Menzer</author>
  </info>
  <r2mfunction>
    <info>
      <r2mversion>0.1beta</r2mversion>
      <author>Fritz Menzer</author>
    </info>
    <expression>sin(5*x*x*y/(x*x*x*x+y*y))</expression>
    <display>
      <width>10</width>
      <linkfunction>
        <x>0</x>
        <y>0</y>
      </linkfunction>
      <linkscreenrel>
        <x>0.5</x>
        <y>0.5</y>
      </linkscreenrel>
```

```
  </display>
</r2mfunction>
<dcxdt>
  <env>10</env>
  <vel>0</vel>
  <key>0</key>
  <ctl>0</ctl>
  <ctlnum>0</ctlnum>
  <envdata>
    <envpoint t="0.000000" val="0.925000" fku="false"/>
    <envpoint t="0.256959" val="0.375000" fku="false"/>
    <envpoint t="0.378302" val="0.050000" fku="true"/>
  </envdata>
</dcxdt>
<dcydt>
  <env>10</env>
  <vel>0</vel>
  <key>0</key>
  <ctl>0</ctl>
  <ctlnum>0</ctlnum>
  <envdata>
    <envpoint t="0.000000" val="-1.000000" fku="false"/>
    <envpoint t="0.107066" val="0.125000" fku="false"/>
    <envpoint t="0.342612" val="0.025000" fku="false"/>
    <envpoint t="0.571021" val="-0.100000" fku="true"/>
  </envdata>
</dcydt>
<ax>
  <env>1</env>
  <vel>0</vel>
  <key>0</key>
  <ctl>0</ctl>
  <ctlnum>0</ctlnum>
  <envdata>
    <envpoint t="0.000000" val="0.000000" fku="false"/>
    <envpoint t="0.021413" val="1.000000" fku="false"/>
    <envpoint t="0.292649" val="0.000000" fku="true"/>
  </envdata>
</ax>
<ay>
  <env>1</env>
  <vel>0</vel>
  <key>0</key>
  <ctl>0</ctl>
  <ctlnum>0</ctlnum>
  <envdata>
    <envpoint t="0.000000" val="0.000000" fku="false"/>
    <envpoint t="0.021413" val="1.000000" fku="false"/>
    <envpoint t="0.292649" val="0.000000" fku="true"/>
  </envdata>
```

```
  </ay>
  <cx0>
    <one>0.8</one>
    <vel>0</vel>
    <key>0</key>
    <ctl>0</ctl>
    <ctlnum>0</ctlnum>
  </cx0>
  <cy0>
    <one>-1</one>
    <vel>1.5</vel>
    <key>0</key>
    <ctl>0</ctl>
    <ctlnum>0</ctlnum>
  </cy0>
  <fx>1</fx>
  <fy>1.005</fy>
</r2msound>
```