Erik Bartmann

Discover modular Synthesizer with

VCV-Rack 2

The installation of the Development Environment for *Vult*

Table of contents

Author	Erik Bartmann
Internet	https://erik-bartmann.de/
Thema	Setting up the development environment for programming plug-ins for the VCV Rack 2 using the Vult programming language.
Version	1.01
Date	19. Januar 2022

The Vult programming

This tutorial is about setting up a development environment under Linux as well as under Windows for the VCV-Rack 2. All necessary steps are mentioned, so that afterwards a simple development of most different plugins for the VCV-Rack 2 can be achieved via the Vult programming language.

Installation steps for Linux

Install Vult compiler

Requirements:

- Installation of node.js
- Installation of npm ($ sudo apt install npm)

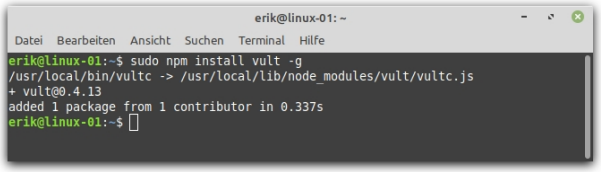

Abbildung 1: Install Vult compiler

Install Rack SDK

Among other things, all versions of the Rack SDK can be found at the following Internet address.

https://vcvrack.com/downloads/

Download the latest version of the Rack SDK for Linux.

Abbildung 2: Rack-SDK

Unzip and install. Note the path to the Rack SDK!

Set path to the Rack SDK in the terminal

The path to the Rack SDK must now be added to, for example, ~/.bashrc or another shell environment, so that it is not necessary to re-enter it every time a terminal is started. The general entry is:

```
export RACK_DIR=<Rack SDK folder>
```

For me the path is:

```
/home/erik/Rack-SDK
```

so that the corresponding entry in ~/.bashrc looks like this:

```
export RACK_DIR=/home/erik/Rack-SDK
```

This modification is best done with a text editor such as *nano*, where the call looks like this:

```
$ nano ~/.bashrc
```

The entry is best placed at the very end of the file, saved with Ctrl-O and the editor is left with Ctrl-X. To test whether this added variable has also been recognized, the Teminal window must be closed and a new one opened. Then enter the following command.

```
$ echo $RACK_DIR
```

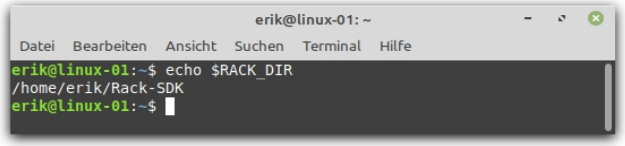

Abbildung 3: Display RACK SDK path

Setting up the development environment

To ensure that the development environment is supplied with all necessary tools, the following command line should be called. (Ubuntu 16.04+):

```
$ sudo apt install unzip git gdb curl cmake libx11-dev libglu1-mesa-dev
libxrandr-dev libxinerama-dev libxcursor-dev libxi-dev zlib1g-dev libasound2-dev
libgtk2.0-dev libgtk-3-dev libjack-jackd2-dev jq zstd libpulse-dev
```

Download Vult Template

To find a suitable entry point, the *RackPlayground* template can be used, which can be found at the following Internet address.

https://github.com/vult-dsp/RackPlayground

For downloading, the following command must be entered into a terminal window, and I have created corresponding folders in the home directory for this purpose. The folder structure looks like this, where I want to save the template in the lowest folder *VCV-Rack*.

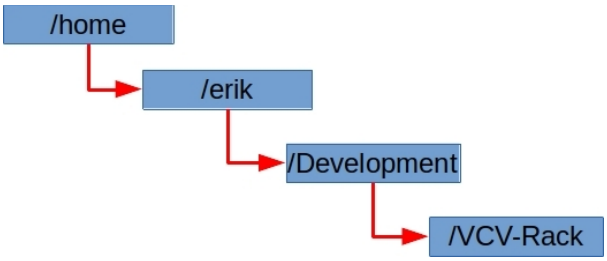

Abbildung 4: Path to the template

From the Git repository, the command

```
$ git clone https://github.com/vult-dsp/RackPlayground
```

now downloads and saves all the necessary files in the following folder.

```
/home/erik/Development/VCV-Rack
```

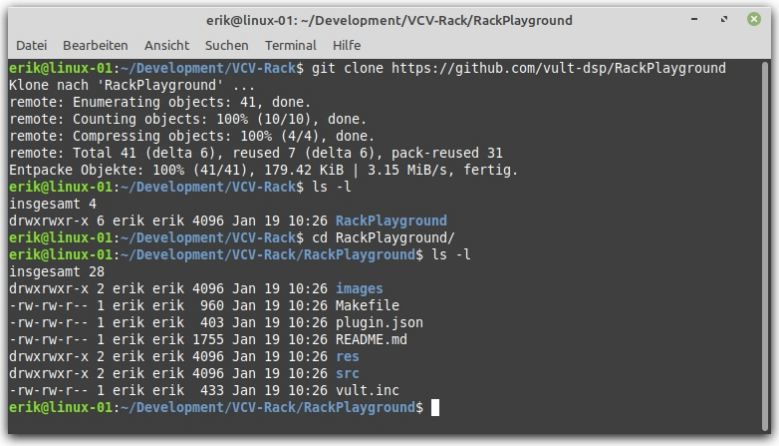

Abbildung 5: Download the template via Git

The Build-Prozess

In the next step, the so-called build process can be started, which ensures that certain dependencies are taken into account during compilation. The make command is a utility for creating and managing groups of programs from a source code. This command requires a so-called Makefile, in which all necessary information is stored. So now I change to the RackPlayground directory, where among other things the mentioned file is located, as you can see in the last screenshot. Now the following command must be executed.

```
$ make -j 4
```

The -j option can be used to specify the number of jobs (commands) that are to be executed simultaneously. The whole thing then looks like this.

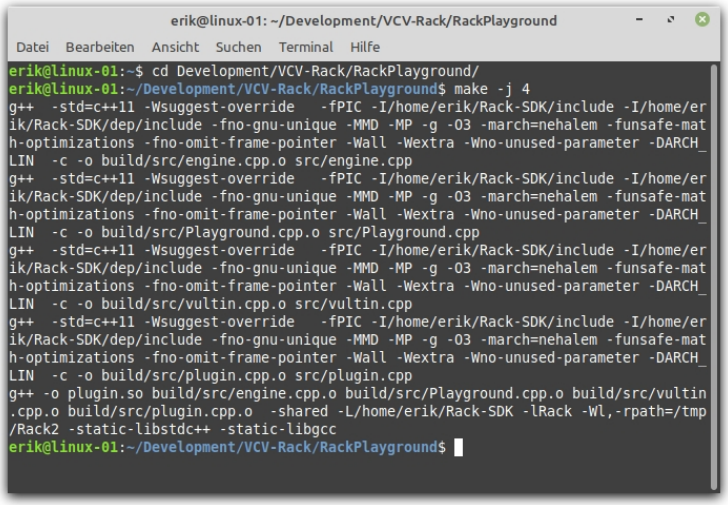

Abbildung 6: The make-Command

To finally install the project, the following command is required.

```
$ make install
```

The result looks like this.

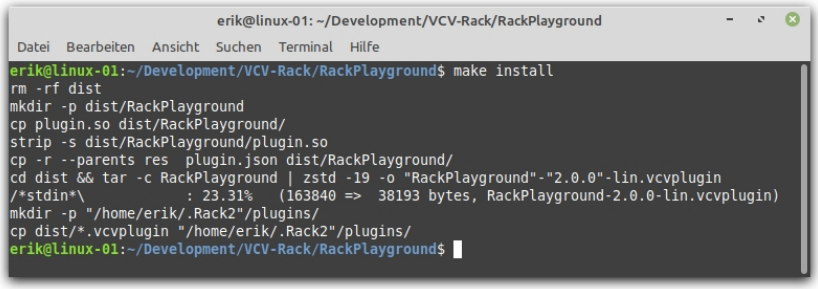

Abbildung 7: The make install-Command

The compilation check

In order to check now whether something has really happened in the file system that can be used as VCV-Rack plugin, a look into a certain directory must be taken. In the terminal window shown last, this path can even be seen in the second last line from the bottom. It says:

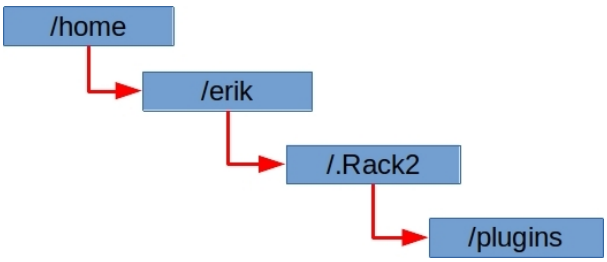

Abbildung 8: Path to the VCV Rack plugins

This is a special directory used by the VCV-Rack installation. After installing VCV-Rack 2 on Linux, all plugins, i.e. extensions, are stored there. Let's take a look there.

There are a lot of extensions in this place, which I have already added to my VCV rack by several subscriptions. Among other things you can also see a special file that I have marked in red on the bottom right of the image.

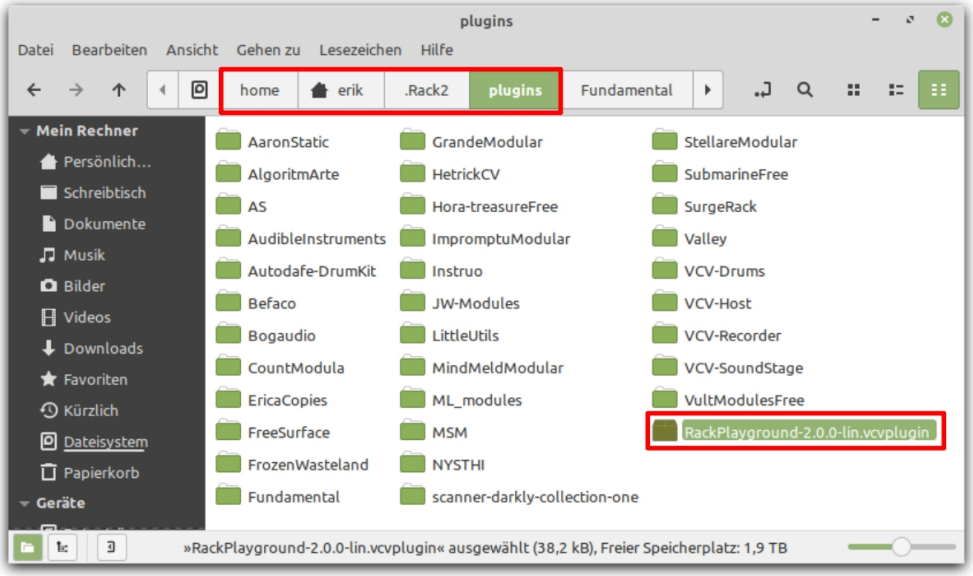

Abbildung 9: The plugin directory of VCV-Rack 2

This is exactly the RackPlayground plugin that has just been created, identified by the file extension vcvplugin. But this extension is different from the other extensions, which are identified by a folder in the file system. Why is that? Quite simply, because after installing such a VCV-Rack extension, the actual installation and conversion to an appropriate subfolder is not done until VCV-Rack is restarted. So I restart my VCV-Rack once and take another look into the plugin directory.

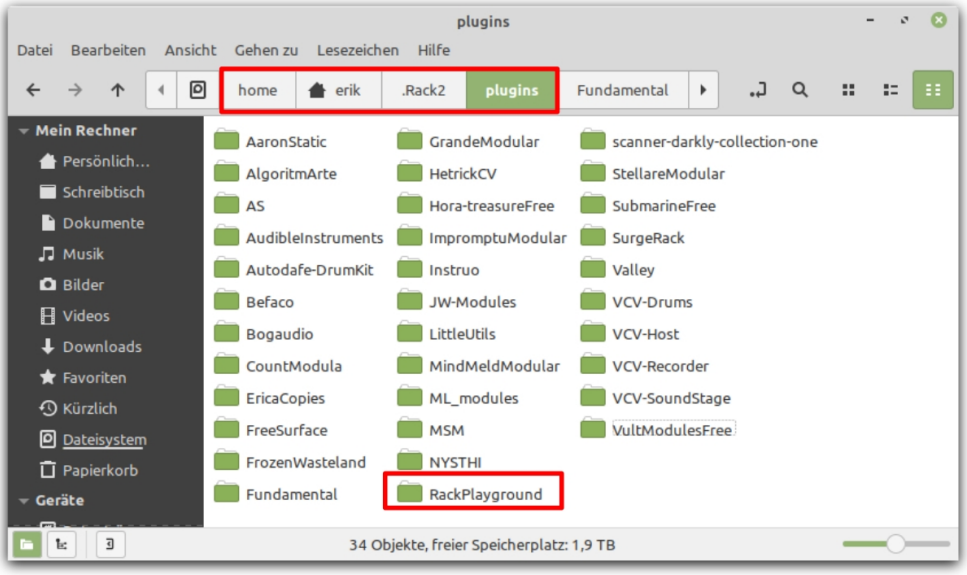

Abbildung 10: The plugin directory of VCV-Rack 2

You can now see that a new subfolder has now been created with the exact name of the RackPlayground template. This now contains all the necessary files to use the plugin. But can the new plugin be found in VCV-Rack? Let's see.

The new RackPlayground plugin in VCV-Rack 2

If the browser is opened in the VCV rack, then the new RackPlayground plugin can be seen directly in the upper left corner and can be taken over and inserted into the VCV rack by a mouse click.

Abbildung 11: The new RackPlayground plugin in VCV Rack 2

Installation steps for Windows

Installing Rack SDK

Among other things, all versions of the Rack SDK can be found at the following Internet address.

https://vcvrack.com/downloads/

Download the latest version of Rack SDK for Windows.

Abbildung 12: Rack-SDK

This file must be unpacked in the file system. The path to it is needed right away so that the compilation process can also find the Rack SDK. For this the environment variable RACK_SDK must be created. I have unpacked the Rack-SDK under *D:\Rack-SDK*.

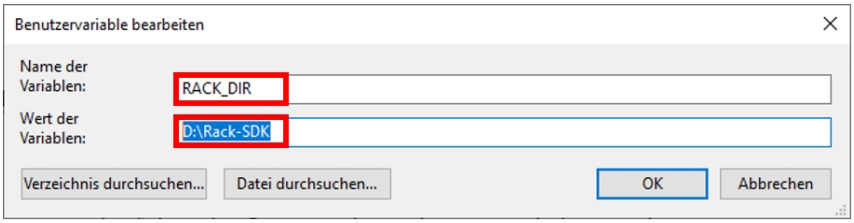

Abbildung 13: The environment variable for the RACK SDK

Install MSYS2

On Windows, we start with the installation of MSYS2. MSYS2 is a collection of tools and libraries that provides an easy-to-use environment for creating, installing and running native Windows software. The software can be found at the following web address, installing the 64-bit version

https://www.msys2.org/

After the installation, various programs are available under Windows, whereby the application marked in red must be used started.

Abbildung 14: The MSYS2 programs

After the call, a terminal window opens, in which the command

```
$ pacman -Su
```

must be entered for a required update, which then looks like this.

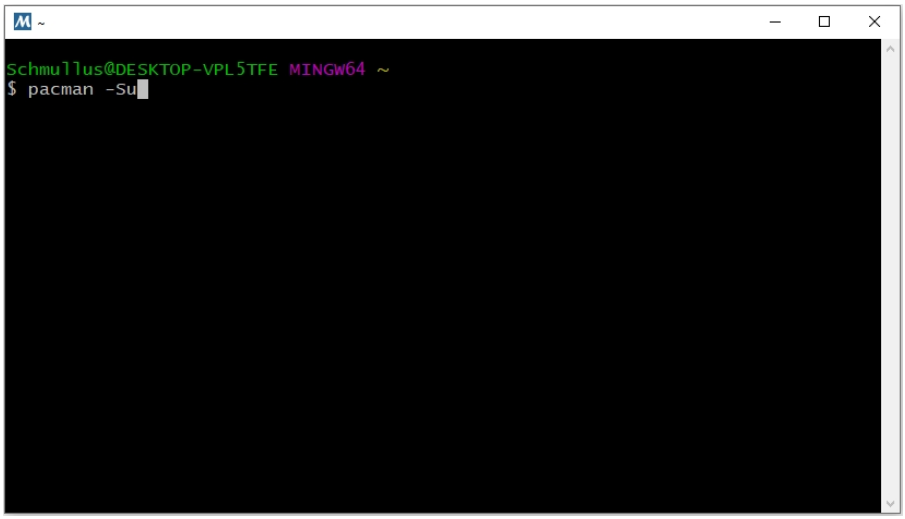

Abbildung 15: The required update is performed

Subsequently, some security questions are asked, which should all be answered with *Y*. The update procedure then begins, with the individual steps being displayed in terms of their progress.

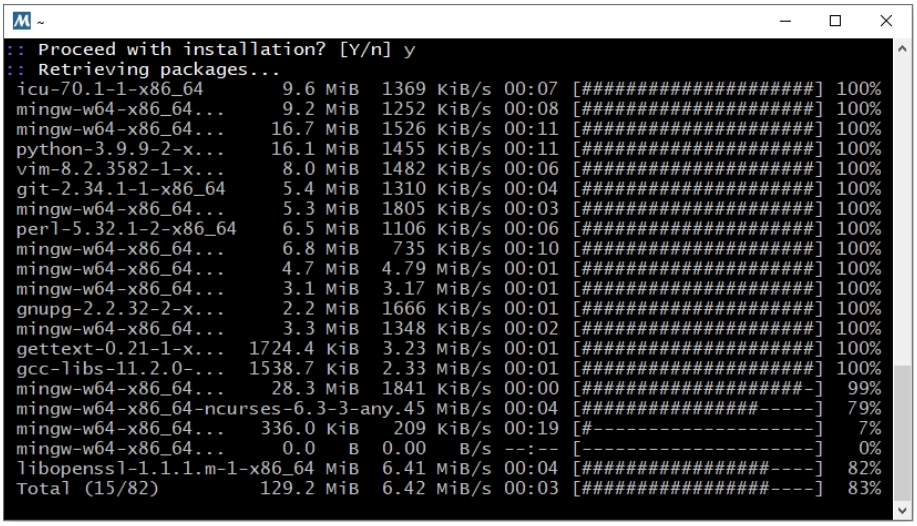

Abbildung 16: The display of the update progress

If the update has been completed successfully, the terminal window must be closed and opened again. The following command line must now be entered.

```
$ pacman -Su git wget make tar unzip zip mingw-w64-x86_64-gcc mingw-w64-x86_64-gdb mingw-w64-x86_64-cmake autoconf automake mingw-w64-x86_64-libtool mingw-w64-x86_64-jq python zstd
```

After some renewed security prompts the installation process starts.

Install Vult compiler

The Vult compiler can be downloaded from the following Internet address.

https://github.com/vult-dsp/vult/releases

The file *vultc.exe* must of course be made known to the system. To do this, the file must be copied to the following directory of the *MSYS2* installation.

```
C:\msys64\mingw64\bin
```

Download Vult Template

In order to download the RackPlayground template also under Windows, Git can be used again, which can be found at the following internet address.

https://github.com/vult-dsp/RackPlayground

To download the data from the Git repository, the following command must be entered into a terminal window.

```
$ git clone https://github.com/vult-dsp/RackPlayground
```

In MSYS2 this looks like this.

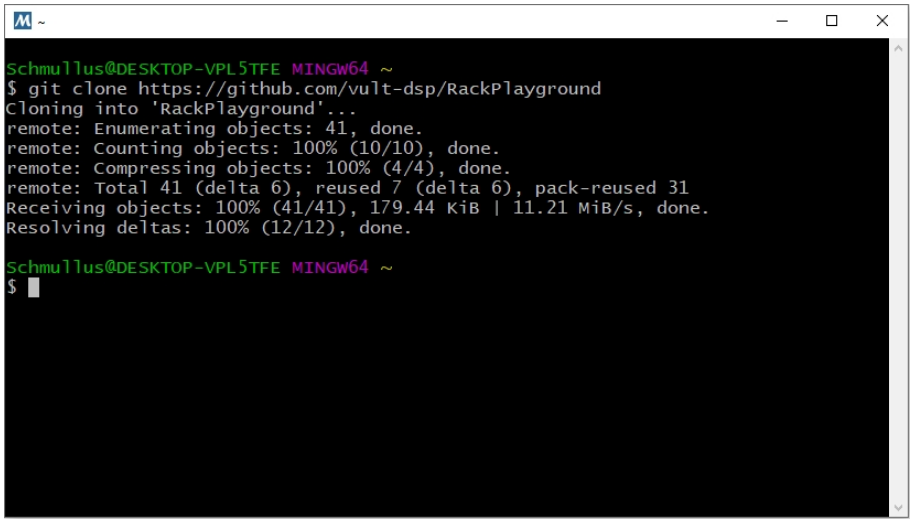

Abbildung 17: Download the template via Git

Now there is the legitimate question where the downloaded files are located in the Windows file system. A look into the following folder provides information.

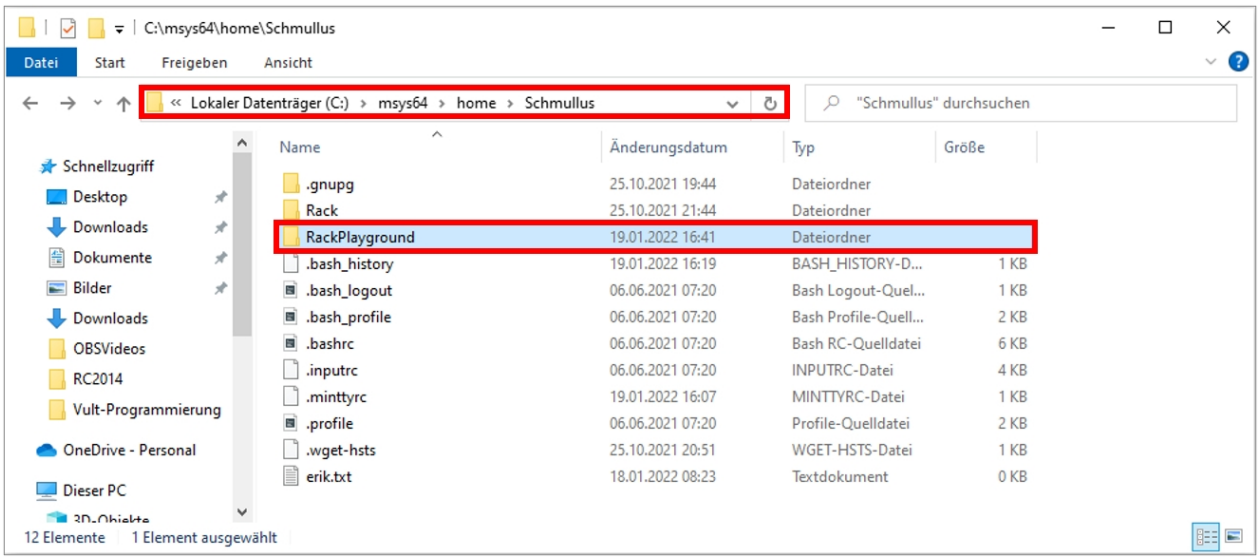

Abbildung 18: The download directory of the template

It can be seen that in the directory

`c:\msys64\home\<User-Name>`

there is a folder called *RackPlayground*. This is the folder from the Git repository. The next step is to update any submodules via Git. This is achieved using the following command line in the MSYS2 terminal window, changing to the template directory beforehand.

```
$ cd RackPlayground
$ git submodule update --init --recursive
```

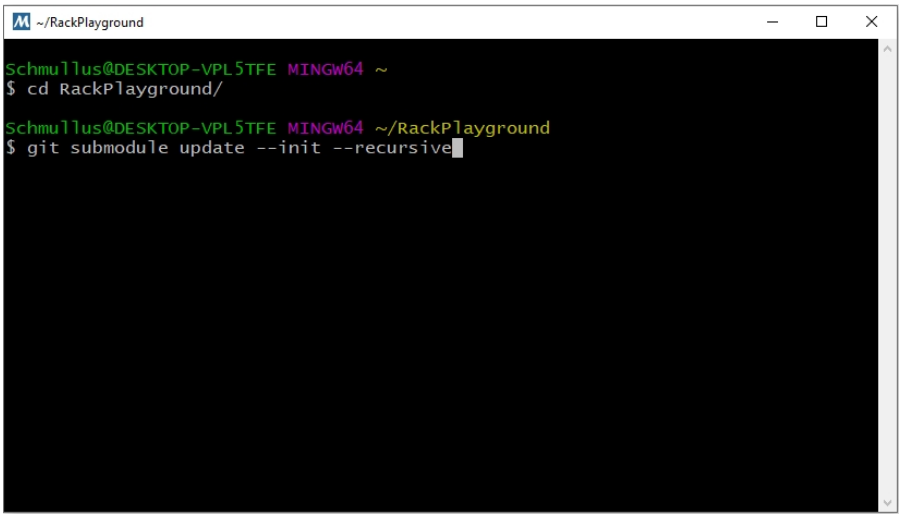

Abbildung 19: An update of the submodules

The build process

In the next step, the so-called build process can be started, which ensures that certain dependencies are taken into account during compilation. The *make* command is a utility for creating and managing groups of programs from a source code. This command requires a so-called *Makefile*, in which all necessary information is stored. The following command must be executed.

```
$ make dep -j 4
```

The -j option can be used to specify the number of jobs (commands) that are to be executed simultaneously. The whole thing then looks like this.

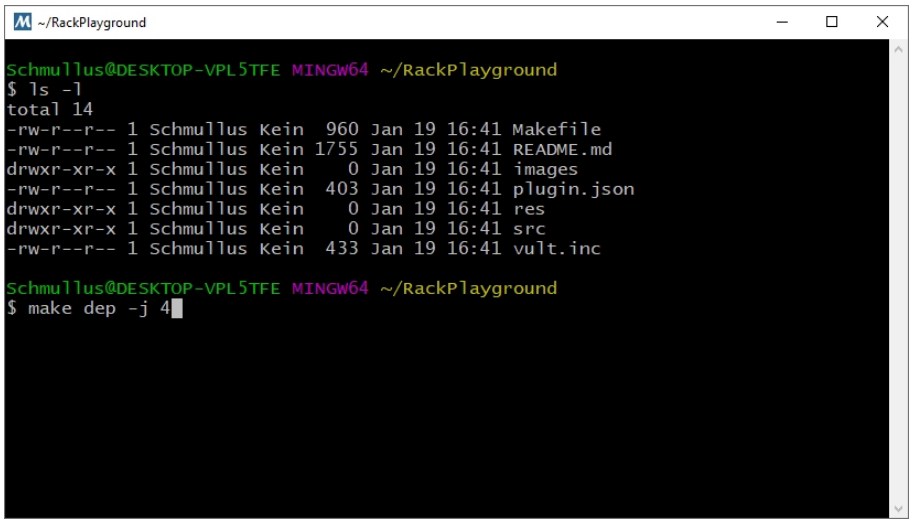

Abbildung 20: The make-dep command

The next step is to call the make command alone.

```
$ make -j 4
```

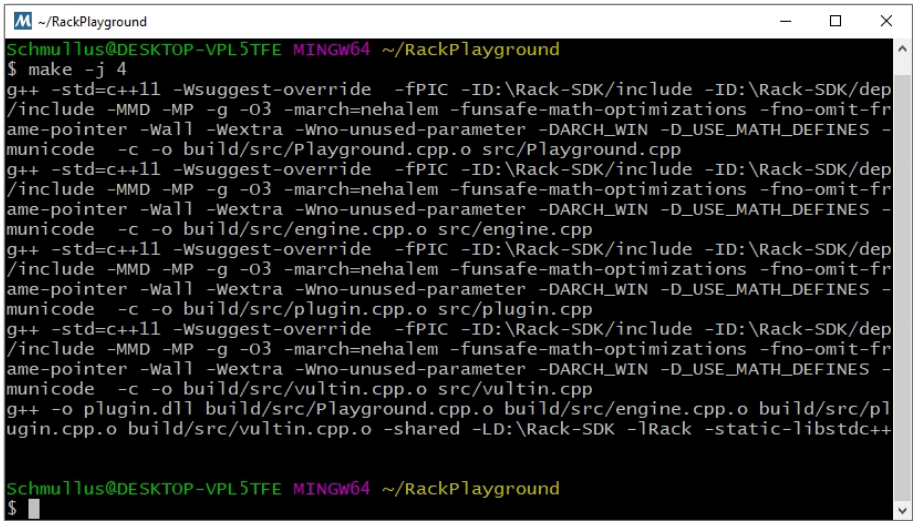

Abbildung 21: The make command

To finally install the project, the following command is required.

```
$ make install
```

The result then looks like this.

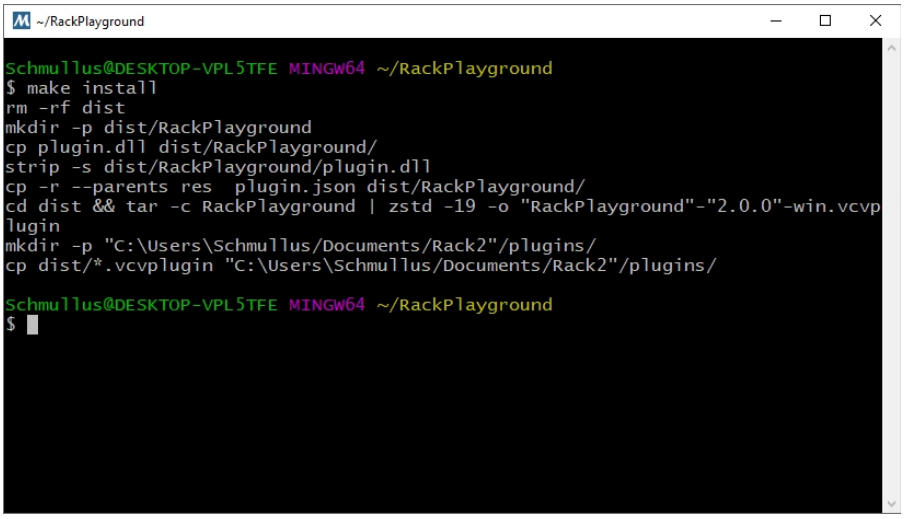

Abbildung 22: The make-install command

The compilation check

To check now whether something has really happened in the file system that can be used as VCV-Rack plugin, a look into a certain directory must be taken. In the terminal window shown last, this path can even be seen in the second last line from the bottom. It reads:

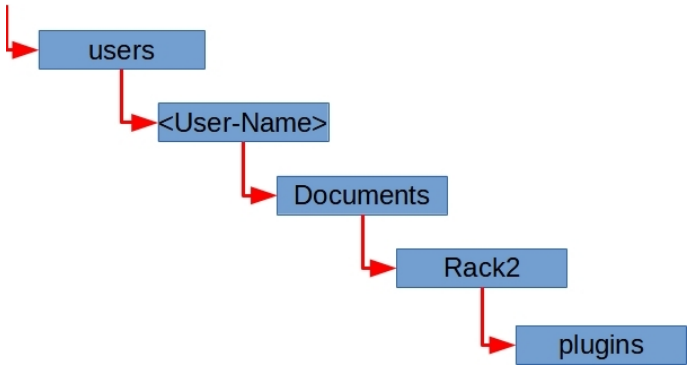

Abbildung 23: Path to the VCV Rack plugins

This is a special directory used by the VCV-Rack installation. After installing VCV-Rack 2 on Windows, all plugins, i.e. extensions, are stored there. Let's take a look inside here. There are a lot of extensions there, which I added to my VCV-Rack by several subscriptions done before. Among other things you can also see a special file, which was marked red in the lower part of the figure.

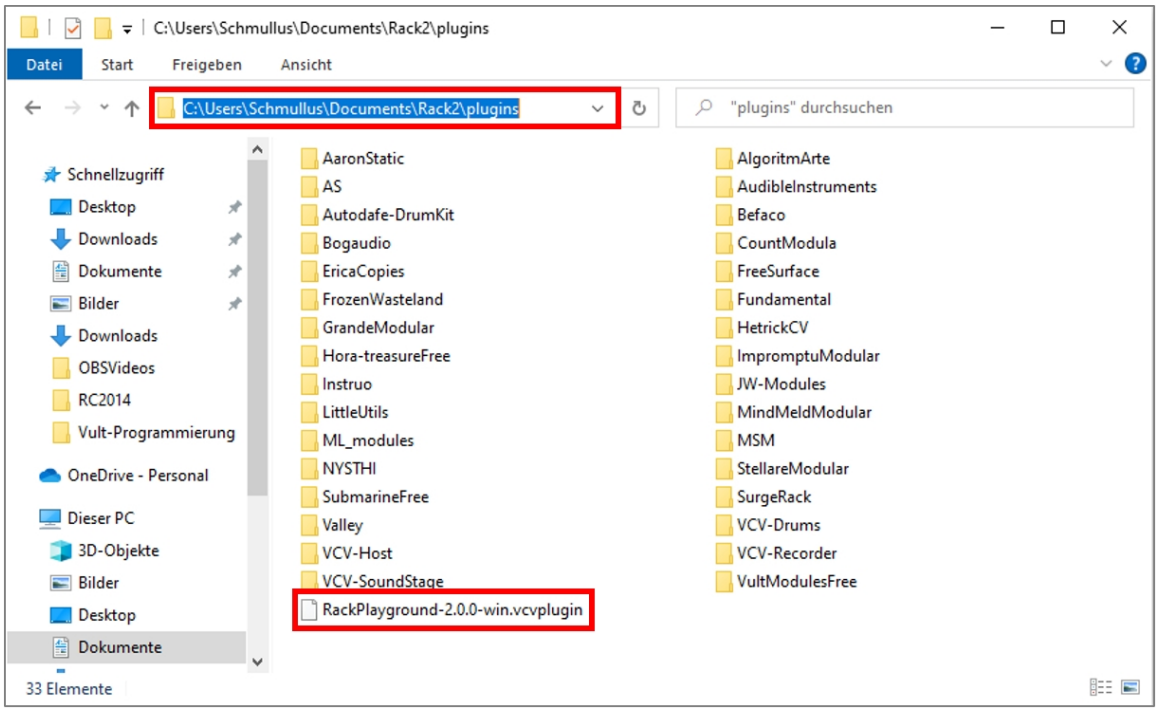

Abbildung 24: The plugin directory of VCV-Rack 2

It is exactly the RackPlayground plugin that has just been created, identified by the file extension *vcvplugin*. But this extension is different from the other extensions, which are identified by a folder in the file system. Why is that? Quite simply, because after installing such a VCV-Rack extension, the actual installation and conversion to an appropriate subfolder is not done until VCV-Rack is restarted.

So I restart my VCV-Rack once and take a look into the plugin directory.

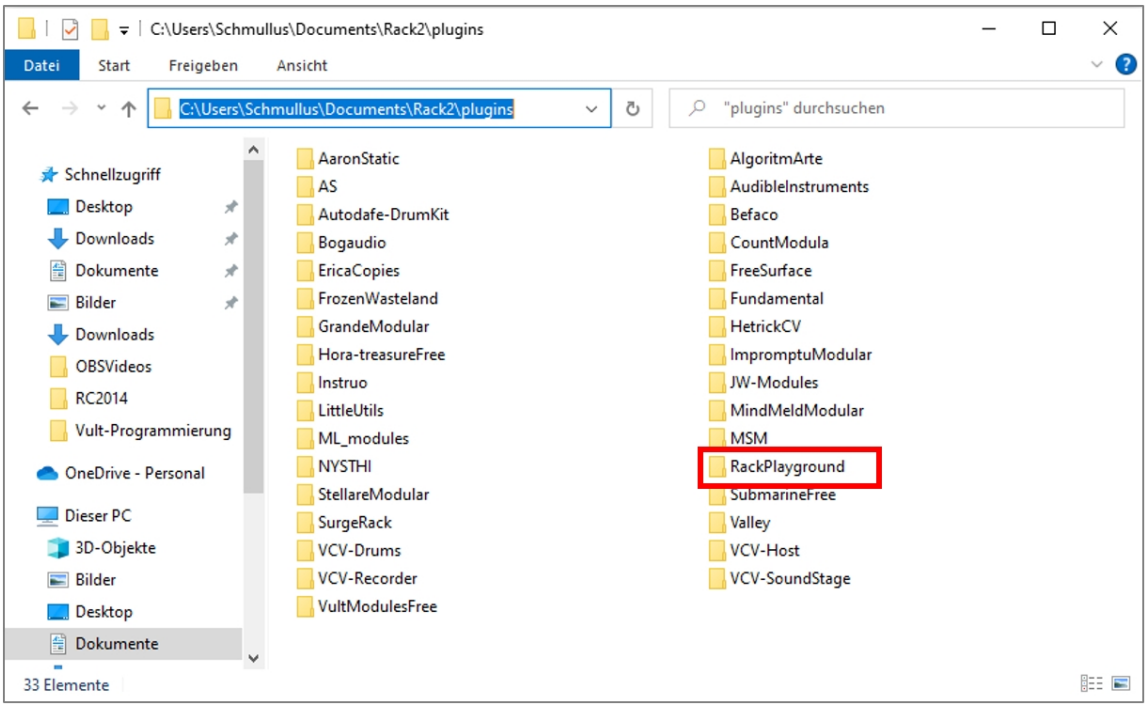

Abbildung 25: The plugin directory of VCV-Rack 2

You can now see that a new subfolder has now been created with the exact name of the RackPlayground template. This now contains all the necessary files to use the plugin. But can the new plugin be found in VCV-Rack? Let's see.

The new RackPlayground plugin in VCV-Rack 2

If the browser is opened in the VCV rack, then the new RackPlayground plugin can be seen directly in the upper left corner and can be taken over and inserted into the VCV rack by a mouse click.

Abbildung 26: The new RackPlayground plugin in VCV Rack 2

Finally

This plugin does not yet have any real functionality and should only serve as a first step to install the development environment accordingly and to show the individual steps required. This should be enough as an introduction to the topic for now, when it comes to preparing the development environment for Vult respectively for the Vult programming language. Via the following link you can get all the necessary information about this topic. Likewise there are numerous videos, which show the work with the programming language in detail.

https://modlfo.github.io/vult/tutorials/

Further information

Of course, all the information can still be found in detail on the following websites.

Erik Bartmann

https://erik-bartmann.de/

https://erik-bartmann.de/?Musik___VCV-Rack

Das VCV-Rack

https://vcvrack.com/

https://vcvrack.com/manual/Building#Setting-up-your-development-environment

Vult

https://modlfo.github.io/vult/tutorials/

https://github.com/vult-dsp/RackPlayground

Have fun coding!

Erik Bartmann