libnmPatch - Nord Modular Patch Library

Table of contents

1. Introduction

libnmPatch implements a C++ API for Nord Modular patch manipulation. It can read and write PCH format files, which are the native fileformat for Nord Modular patches.

The API handles both the static and dynamic patch state. The static state is read and written to file, and is also the state that is stored in the synthesizer memory banks. The dynamic state is updated by an external agent (libnmProtocol) connected to a real synthesizer and consists of lights and meters.

The library can also calculate patch resource allocation.

2. Patch Class Diagram

3. Creating Patches from Scratch

Create a new patch with the default constructor.

```
#include <nmpatch/patch.h>

Patch patch();
```

Set some basic patch parameters.

```
patch.setName("Pling");
patch.setPortamento(Patch::NORMAL);
```

Add an AudioIn and a 4Output module to the polyphonic section. If the second parameter to addModule() is skipped, the module will be assigned the next available index.

```
#include <nmpatch/modulesection.h>
#include <nmpatch/module.h>

ModuleSection* polySection = patch.getModuleSection(ModuleSection::POLY);
Module* input  = polySection->newModule(2);
Module* output = polySection->newModule(3);
```

Connect a couple of cables between the modules.

```
#include <nmpatch/cable.h>
#include <nmpatch/moduletype.h>

Cable* left = polySection->newCable(Cable::RED,
```

```
                                        output->getIndex(), 0,
ModuleType::INPUT,
                                        input->getIndex(), 0,
ModuleType::OUTPUT);
Cable* right = polySection->newCable(Cable::RED,
                                         output->getIndex(), 1,
ModuleType::INPUT,
                                         input->getIndex(), 1,
ModuleType::OUTPUT);
```

The patch is finished.

4. Reading and Writing Patch Files

A patch is loaded from file when a filename is given to the Patch constructor.

```
#include <nmpatch/patch.h>

Patch pling("/home/andersson/patches/pling.pch");
```

A patch cat be serialized to a string. It is then up to you what you want to do with it, e.g. write it to a file.

```
string buffer = pling.write();
```

The API is not consequent here, I know. You are welcome to change it if you know how to get flex/bison to parse a string instead of a file.

5. Module Information and Mappings

6. Architecture