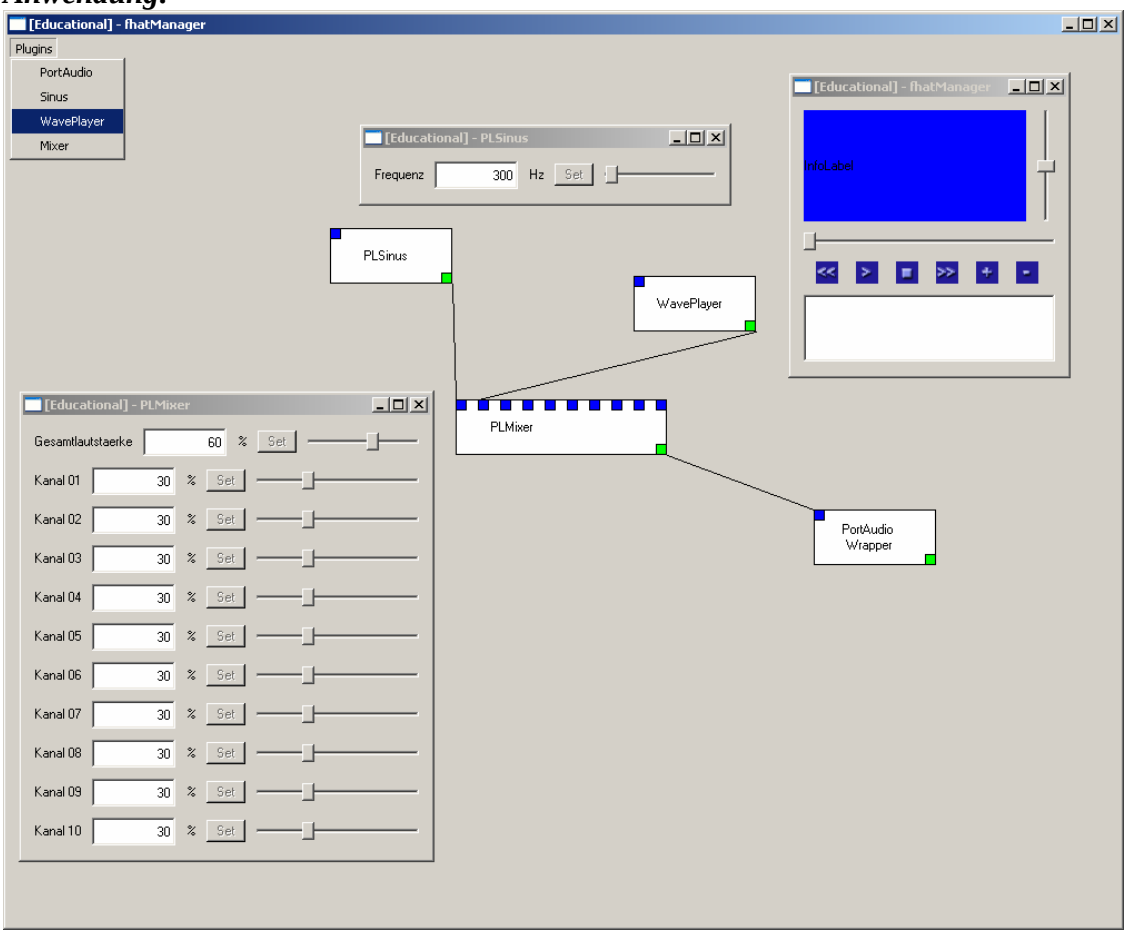
fhatManager…

…eine Applikation zum Verknüpfen von Plugins.

Anwendung:



- Plugins werden durch Auswählen im Menü instanziert. Mit gedrückter linker Maustaste können sie auf der Arbeitsfläche verschoben werden.
- Verbinden: Mit gedrückter rechter Maustaste vom liefernden zum beziehenden Plugin ziehen:
- Verbindung lösen: Rechtsklick auf beziehendes Plugin.
- GUI eines Plugins öffnen: Doppelklick auf Plugin

Aufbau:

Die Klasse **FhatManager** ist von QMainWindow abgeleitet. Sie verwaltet die Plugins mit dazugehöriger GUI und stellt die GUI zum Verknüpfen der Plugins zur Verfügung. Daneben gibt es die Klasse **PluginHandler**. Sämtliche Methoden und Membervariabeln dieser Klasse sind statisch. **FhatManager** bezieht von **PluginHandler** Informationen über die zur Verfügung stehenden Plugins, und ruft die create-Methoden zum Instanzieren von Plugins und GUIs auf.

Hinzufügen von Plugins:
Nur das File *PluginHandler.h* muss angepasst werden:

```
17  class PluginHandler
18  {
19  private:
20      enum Plugin {ePLPortAudio, ePLSinus, ePLWavePlayer, ePLMixer};
21      /**/
24  public:
25      static PluginAbstract* createPlugin(unsigned int plugin)
26      {
27          PluginAbstract* ret=0;
28          switch(plugin) {
29              case ePLPortAudio:  ret= PLPortAudio::getInstance(100); break;
30              case ePLSinus:      ret= new PLSinus(300); break;
31              case ePLWavePlayer: ret= new PLWavePlayer(); break;
32              case ePLMixer:      ret= new PLMixer(); break;
33          }
34          return ret;
35      }
36      /*************************************************************************/
37      static GuiPluginAbstract* createPluginGUI(unsigned int plugin, PluginAbstract* FXPlugin)
38      {
39          GuiPluginAbstract* ret=0;
40          switch(plugin) {
41              case ePLPortAudio:  ret= new GuiPluginPortAudio(reinterpret_cast<PLPortAudio*>(FXPlugin)); break;
42              case ePLWavePlayer: ret= new GuiPluginWavePlayer(reinterpret_cast<PLWavePlayer*>(FXPlugin));  break;
43              default:        ret= new GuiPluginGeneric(FXPlugin); break;
44          }
45          return ret;
46      }
47      /*************************************************************************/
48      static std::string getPluginName(unsigned int plugin)
49      {
50          std::string ret;
51          switch(plugin) {
52              case ePLPortAudio:  ret= "PortAudio"; break;
53              case ePLSinus:      ret= "Sinus"; break;
54              case ePLWavePlayer: ret= "WavePlayer"; break;
55              case ePLMixer:      ret= "Mixer"; break;
56              default:            ret= "no valid plugin"; break;
57          }
58          return ret;
59      }
60
61      /**/
64      const static int iNumPlugins=4;
65  };
```

(Zeilennummern beziehen sich auf das Bild.)

- Includes für Plugin und eventuell GUI hinzufügen
- Zeile 20: hinzufügen in der Enumeration. Namenskonvention: „e“+Klassenname
- Zeile 28ff: einen neuen case einfügen, nachdem in *ret* ein Zeiger auf ein neu generiertes Plugin steht
- Zeile 40ff: einen neuen case einfügen, nachdem in *ret* ein Zeiger auf ein neu generiertes GUI steht (Nur wenn das Plugin ein spezielles GUI hat! Sonst wird bei *default* ein generisches GUI instanziert.
- Zeile 51ff: einen neuen case einfügen, nachdem in *ret* der Name des Plugins steht
- Zeile 64: *iNumPlugins* um 1 erhöhen

Ausblick:
Das Programm eignet schon gut zum Testen von Plugins, ist aber noch sehr rudimentär. Nachfolgende Versionen werden wohl noch weitere Features implementieren (z.B. Konfigurationen speichern). Auch hinsichtlich Error-Handling gibt es noch einiges zu tun bis fhatManager das Zertifikat „dauProof“ erhält.