VOID AUDIO Version Number v.0.2.0.0

iPlug2OOS VOID Audio Fork

Derek Wingard

Version Date: 9/11/2025

Abstract Added AI agent with tasklist functionality.

Keywords AI agent

Version Date: 9/11/2025
Abstract Added AI agent with tasklist functionality.
Keywords AI agent

1 Version `v.0.2.0.0` Update List

1.1 Key Features Added

Brief list of all the added functionality

1.1.1 AI Agent

You can now configure AI agent tasks in `agent_tasks/`. It is recommended you label them by the convention `Number_Description` so you can control the order of the tasks. The task configuration files follow the same outline of `PATH>QUERY>ASSOCIATED`:

```
PATH = "TemplateProject/TemplateProject.h"
QUERY = """
Given this header file, update it so that it also includes a controllable parameter for mixAmount and width
"""
ASSOCIATED = ""
```

Or with ASSOCIATED:

```
PATH = "TemplateProject/TemplateProject.cpp"
QUERY = """
Given this cpp file, update it so that it also includes the controllable parameter for mixAmount and width spe
"""
ASSOCIATED = "TemplateProject/TemplateProject.h"
```

D. Wingard
VOID Audio
E-mail: fromthevoid.audio@gmail.com

The files that it overwrites are stored at `agent_backups/` and can all be restored using:

```
./restore.py
```

2 Version `v.0.1.0.0` Update List

2.1 Key Features Added

Brief list of all the added functionality

2.1.1 `init.sh`

This script is for setting up the project once the Codespace is started. Edit the configuration file `config.txt` to specify the project:

```
PROJECT_NAME = MyPlugin

MANUFACTURER_NAME = MyCompany
```

with more specification to the build expected to come in later versions of this script. Then, activate the init script with:

```
./init.sh
```

2.1.2 `setup.sh`

Rebuilt the `setup.sh` script that used to exist in the repo. It now gets called from `init.sh` and is fed arguments from `config.txt`.

2.1.3 undo_setup.sh

Makes the process a little more foolproof. You can now undo the `init.sh` script and reset the project folders/-files to `TemplateProject`. Reads correct `PROJECT_NAME` from `config.txt`. The folders `.github/workflows` and `.vscode` are `git` restored so make sure you have the initial state you want to revert to as your previous push or else you will have to manually delete and copy back in the template files. If you have your tracking set up, then simply reset the project with:

```
./undo_setup.sh
```

2.1.4 Updated duplicate.py

The `duplicate.py` script was changing my custom scripts so I updated it to no longer touch them.

2.1.5 Added LaTeX functionality and Docs to the Codespace

Now includes scripts to setup LaTeX to compile the README.tex file. Edits can be made at `VOID_Docs/svjour/README.tex`. When you are ready to compile them, first use:

```
./VOID_Docs/setup_latex.sh
```

then you will be able to compile with

```
./VOID_Docs/tex.sh
```

Compiles `README.pdf` to `VOID_Docs`.

2.1.6 Optimized Git Commands

I added my personal `git` commands for quality of life stuff. Updated `devcontainer.json` so they are available instantly when the workspace is built.

`gitster`:

```
------------------------------
read -p "Commit message: " msg
git add .
git commit -m "$msg"
git push origin master
------------------------------
```

`gitmain`:

```
------------------------------
read -p "Commit message: " msg
git add .
git commit -m "$msg"
git push origin main
------------------------------
```

`gitbranch`:

```
------------------------------
read -p "Branch name: " branch
read -p "Commit message: " msg
git add .
git commit -m "$msg"
git push origin "$branch"
------------------------------
```

3 Expected Next Steps

The ideas we have of where this project will be going.

3.1 AI Agent

More functionality to maybe include some kind of git-style merge mechanism or even have it run through its own branch and initiate merges so the inclusion is more controlled. Could even set up other agents to manage merges and whatnot. Many such ideas.

3.2 `init.sh` Script

As it stands, learning how to use this framework as effeciently as possible will involve improvements to this init build script. Ideally, we will have a set of specifications available in `config.txt` that can get started on specific build types that require unique logic, for example:

```
REQUIRES_REALTIME_DISPLAY = bool
IS_INSTRUMENT = bool
NUM_PARAMS = int
BUILD_AAX = bool
LICENSING_TYPE = string
EXTERNAL_LIBS = string
etc...
```