Revolutionizing Audio Development with GitHub CoPilot: The AI-Powered Assistant for Streamlining Code

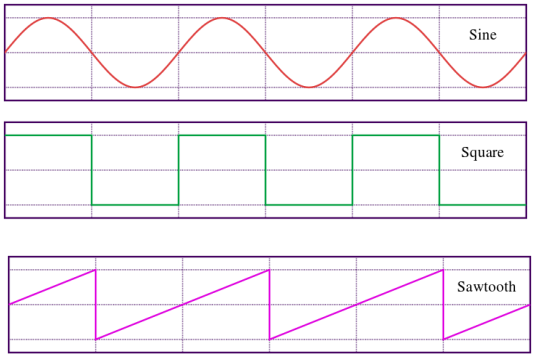
By: Patrick Lacey

Artificial Intelligence has been making big strides recently, and there's a relatively new product called GitHub CoPilot that has caught my attention. Due to the nature of this tool, it has potential to help me code faster and more efficiently. I decided to test it out while coding a simple but novel panning plug-in. At the same time, I was giving CoPilot a try I also took a quick look at using IntelliSense and at using ChatGPT through a web browser to do some of the same coding tasks.

My plug-in processes the panning in a couple different ways depending on the state of two buttons on the top left as described in the logic table below:

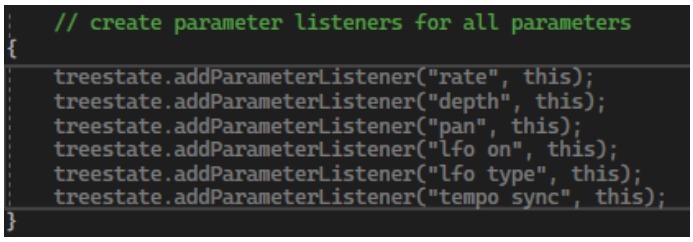
BPM Sync	LFO	Active Controls	Notes
	X	Pan slider	
X		Depth knob Rate knob	
X	X	Depth knob	Uses BPM reported by the DAW for the rate

Using the DSP class's Oscillator object and an audio buffer, I created a low frequency oscillator with different waveform options to put its output into the audio buffer. From there, I made the panning value of my output AudioBlock index equal to the depth multiplied by the value of the LFO audio buffer's index. After that block is processed, the audio buffer needs to be cleared so it can be filled with the continuing samples that the LFO is generating.



The plug-in has three different waveform options that create different movements in panning. The sine waveform is very smooth in moving back and forth from left to right, while the square waveform is very sharp with its jumps between hard left and right, and the Saw waveform creates this typewriter effect where it travels from left to right smoothly and then jumps back to hard left. I chose these three waveforms specifically because these are the three common panning movements, I hear that are controlled by an LFO. For a deeper understanding of the plug-in, take a look at the source code I've provided in my GitHub repository.

Now let's talk about the AI tools I used for this project and their functionality. GitHub CoPilot is called an "AI pair programmer" which essentially an AI extension inside your IDE. I downloaded it on Visual Studio 2022, linked my GitHub account, and I was on my way to faster coding. Even though it uses the same AI model that ChatGPT does I quickly realized the convenience of being able to access AI features without tabbing between the IDE and a ChatGPT session running in the browser. I initially started the creation of my AutoPan header file using ChatGPT, which was helpful because it typed out a page of a well-structured class object. CoPilot takes instruction through commenting a request, but it's more limited in response length and focused more on feeding predictive code blurbs as you go, such as this:



There already is a predictive code AI feature provided by Visual Studio called IntelliSense which is helpful. It helps fill out the correct argument list for a function, or get to the correct member data in an object, but it only has code-completion functionality, which means its focused on the line of code being typed, while CoPilot can supply multi-line suggestions. Also in regards to audio development using JUCE, I found that CoPilot's pretrained model is more helpful since they pull from GitHub public repositories.

I found using CoPilot to be very beneficial and I plan on continuing to use it, but it's not perfect. There were many times where it appeared that it was pulling code directly from another project due to the naming conventions being different. There were also many times where the code supplied was logically correct, but it didn't follow the JUCE framework labelling and created code that was harder to follow and debug. I believe that with better training models for AI in the short future these problems will be taken care of, but for now coding with AI tools still requires human editing and debugging.

Overall, my initial observation is that GitHub's CoPilot is the most effective tool for programming. Since the learning models are still improving, the potential for changing the way we all code is huge. The feature that I've found to be the most important is that you don't have to even leave your IDE, which boosts efficiency and keeps your mind in that flow state. Essentially boosting whatever coding language you're using to a higher-level language.