Using Slot Local Variables

If a program is loaded in several slots, the state of its parameters (e.g., level, cutoff, resonance, etc.) is synchronized across these slots. The same is true for the state of global variables in a Lua script. If a global variable is changed in one slot, its state will change in every other slot where the program is loaded.

However, the engine's playback state of the program is handled independently for each slot. This is necessary because each slot can receive different MIDI events. If some functions in your script depend on global variables that store the state of MIDI events, the automatic synchronization of global variables is usually a hindrance, because the global variables will be overwritten by the slot that received the latest MIDI events. To attain global variables that operate independently per slot, use defineSlotLocal.

On this page:

Related pages:

- defineSlotLocal

Declaring Slot Local Variables

You declare slot local variables by calling defineSlotLocal with the name of the corresponding global variable as argument. You can call defineSlotLocal before or after the initialization of the variable, but it is common practice to call the function in advance.

Example

The following example plays a classic up arpeggio. The variables for the arpeggio (noteBuffer, arpRunning and arpeggioNotes) need to be declared as slot local variables, otherwise, the script will not work as expected if the program is loaded into more than one slot.

To explore the script:

1. Download SlotLocalVariables.vstpreset.
2. Load the program twice into the **Slot Rack** and send different chords to the slots.
3. Comment out the declaration of the slot local variables and send different chords to the slots again.

If slot local variables are declared, both slots play separate arpeggios. If slot local variables are not declared, only one slot will play an arpeggio with a mix of the chords that you are sending to the slots.

```
--[[
    Simple arpeggiator playing 1/16 notes with fixed velocity.
    The global variables noteBuffer, arpRunning and arpeggioNotes use defineSlotLocal.
--]]

-- global statements

-- declare slot local variables
defineSlotLocal("noteBuffer")
defineSlotLocal("arpRunning")
defineSlotLocal("arpeggioNotes")

-- initialize variables
-- set note length to 1/16, i.e., one beat divided by four
noteLength = 0.25
-- set a fixed velocity of 100
fixedVelocity = 100
-- buffer for held notes
noteBuffer = {}
-- playback state of arpeggiator
arpRunning = false

-- transfer the held notes into an array and sort its values
function sortNotes()
    arpeggioNotes = {}
    for note in pairs(noteBuffer) do
        arpeggioNotes[#arpeggioNotes+1] = note
    end
    table.sort(arpeggioNotes)
end

-- play up arpeggio
function playArpeggioUp()
    arpRunning = true
    -- start playback with index 1
    local i = 1
    -- playback for as long as there are arpeggio notes
    while arpeggioNotes[i] do
        local id = playNote(arpeggioNotes[i], fixedVelocity, 0)
        wait(getBeatDuration() * noteLength * 0.5)
        -- release the current voice after half the note length
        releaseVoice(id)
        wait(getBeatDuration() * noteLength * 0.5)
        -- increase index by 1 to play the next note
        i = i + 1
        -- reset playback to index 1, if index out of range
        if not arpeggioNotes[i] then
            i = 1
        end
    end
    arpRunning = false
end

function onNote(event)
    -- write notes to the buffer
    noteBuffer[event.note] = event.velocity
    sortNotes()
    -- start the arpeggio only if it is not running
    if arpRunning == false then
        playArpeggioUp()
    end
end

function onRelease(event)
    -- remove notes from the buffer
    noteBuffer[event.note] = nil
    sortNotes()
end
```