SuperCollider (SC)

Beginner’s guide workshop

kauewerner.github.io

What is SC?

Class -> Object -> Instances

Object-oriented language and tool for Sound synthesis and Digital Signal Processing (DSP)

Open Source!!!

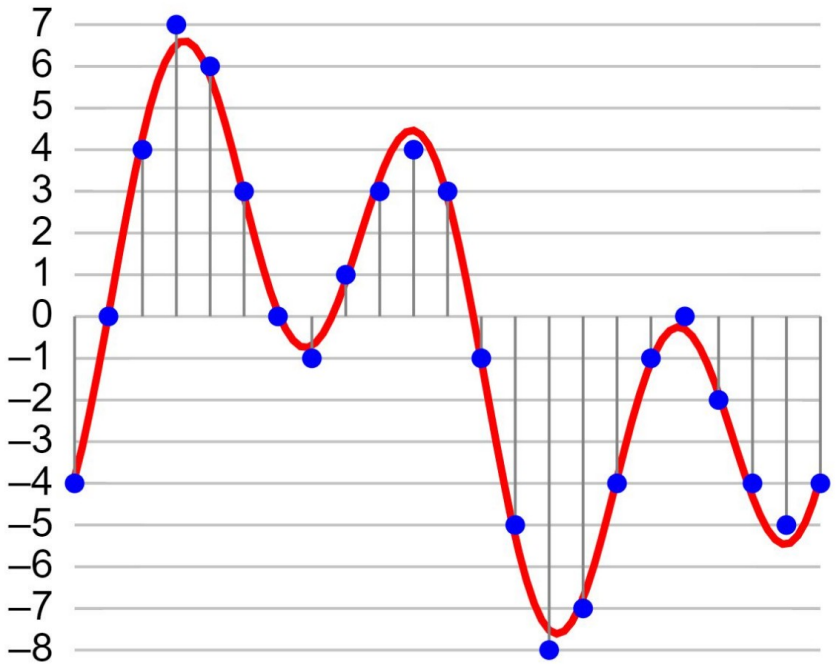
https://supercollider.github.io/downloads.html

Sound and digital audio

- Sound information can be **generated/recorded/reproduced** via amplitude oscillations (or waves)

- Digital audio is one of the methods to represent and/or stored this information

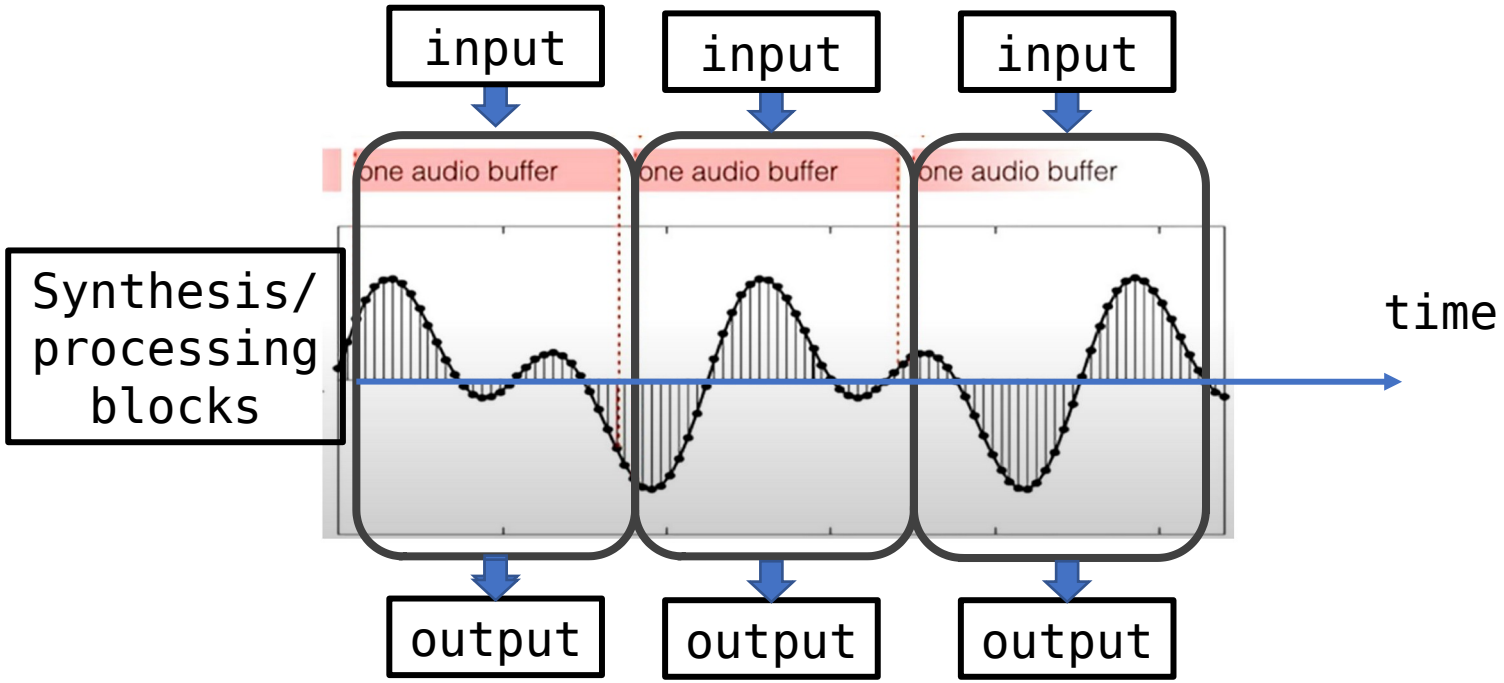
- The **sound wave** is discretized by **samples** of data



Digital synthesis and processing

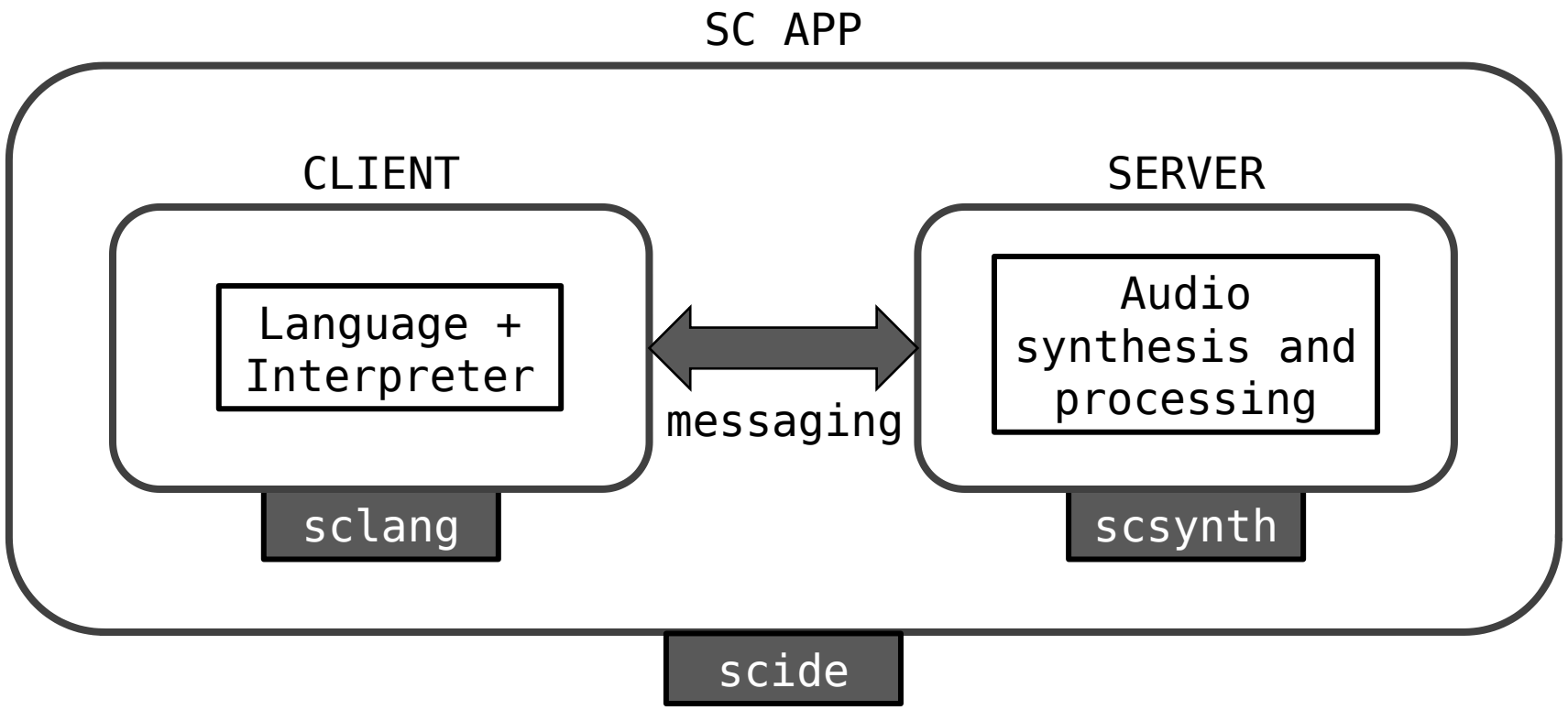
- **Synthesis**: generation of the samples to form a specific wave
- **Processing:** alteration of the samples from an already existing wave

- In performances, these processes are done in **real time**, which occur in chunks of audio data (also known as buffers)



How does SC work?

- Client-Server architecture



Integrated Development Environment (**scide**)

- 3 main parts
  - Scripting
  - Post window (debugging, errors, warnings)
  - Help browser (Documentation)

- General running key combinations
  - Shift + Enter/Return (line)
  - Ctrl/Cmd + Enter/Return (blocks of code)

Audio server (**scsynth**)

- Lean and efficient command line program dedicated to audio synthesis and processing.

- It knows nothing about SC code, objects, Object Oriented Programming, or anything else to do with the SC language, only **OSC messages**

  Open Sound Control

- While synthesis is running, new modules can be created, destroyed and repatched

SuperCollider language (**sclang**)

- Compile the scripts to **OSC messages** to be sent to the audio server (**scsynth**):
  - “*The user writes poetry (so to speak) in the SuperCollider language which is then paraphrased in OSC prose by the sclang interpreter, to be sent to the server*”

SuperCollider language (**sclang**)

- Variables
  - Global
    - All single lowercase letters
    - Any string starting with (~ + lowercase)
  - Local
    - var …

- Classes
  - Always beginning with UPPERCASE letters

SuperCollider language (**sclang**)

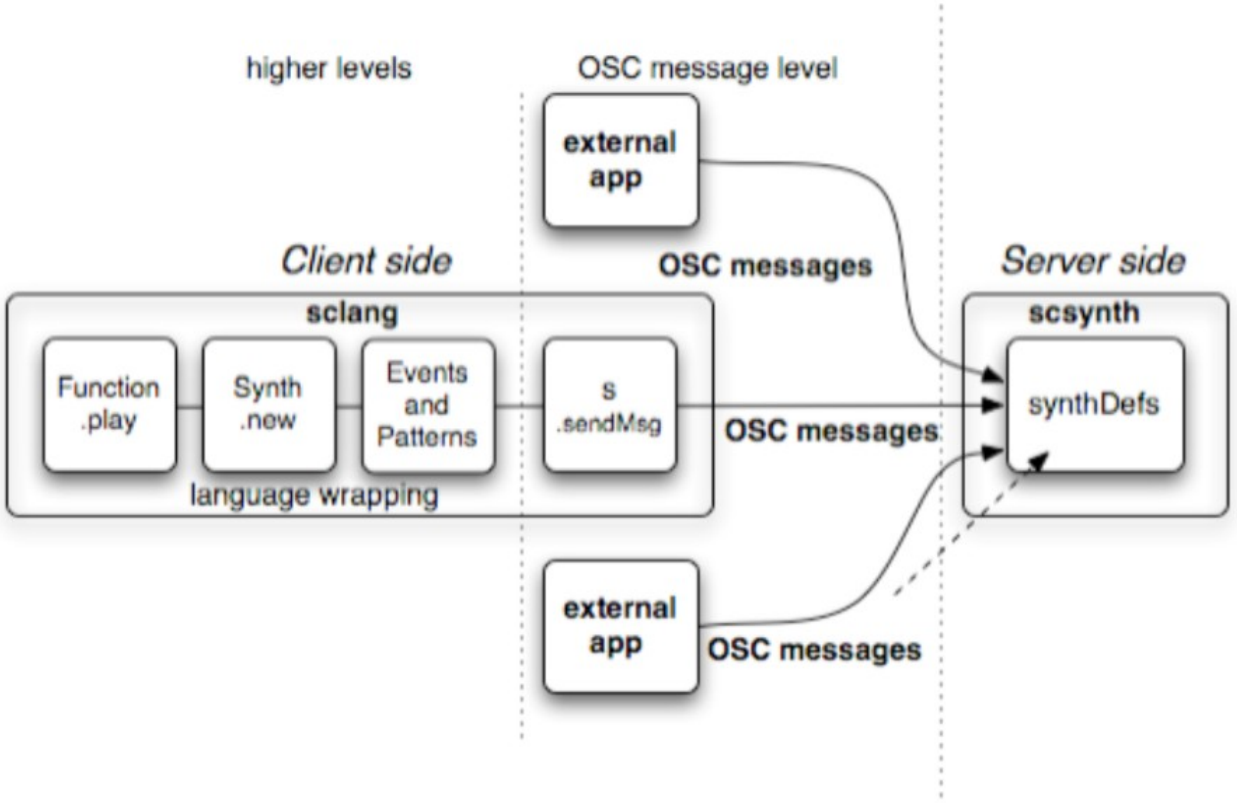


Figure 3. Sclang as a high-level client

General server-related controls

- Start/Restart the server
  - s.boot / s.reboot
- Quit/Kill the server
  - s.quit
- Monitoring
  - s.meter
  - s.plotTree
  - s.scope
- Free/destroy all running synth sounds
  - Ctrl/Cmd + .

Unit Generators (UGens)

- UGens represent calculations with signals
- They are the basic building blocks of synth definitions on the server
- Used to generate or process both audio (**.ar**) and control (**.kr**) signals
- SuperCollider has more than 1000?… and counting…

Unit Generators (UGens)

- These are the main categories of UGens:
  - sources: periodic, aperiodic
  - filters
  - distortion
  - panning
  - reverbs
  - delays and buffer UGens
  - granular synthesis
  - control: envelopes, triggers, counters, gates, lags, decays
  - spectral

SynthDefs and Synths

- Classes vs Instances
- **SynthDefs** are synth definitions which mean data about UGens and how they're interconnected
- **Synths** on the server are basically just things that make or process sound, or produce control signals to drive other synths

Patterns

- Patterns are one of the most powerful elements of the SuperCollider language
- Patterns describe calculations without explicitly stating every step
- They are best for tasks that need to produce sequences, or streams, of information
- Patterns define behavior; streams execute it

Further exploration (next workshops?)

- Buffers
- Audio/Control Bus
- MIDI messages (note/control)
- Creating a GUI
- Quarks/extensions

Where to look for help and explore?

- SC Help (sometimes not enough but very good!) also available online (https://doc.sccode.org/)
- https://scsynth.org/ -> main discussion forum
- https://sccode.org/ -> where people share their code

- Eli Fieldsteel channel
  - Tutorial series (“short” and fast)
  - Longer lectures