The Lab Streaming Layer – Introduction and Overview

Christian A Kothe, CTO

The Lab Streaming Layer

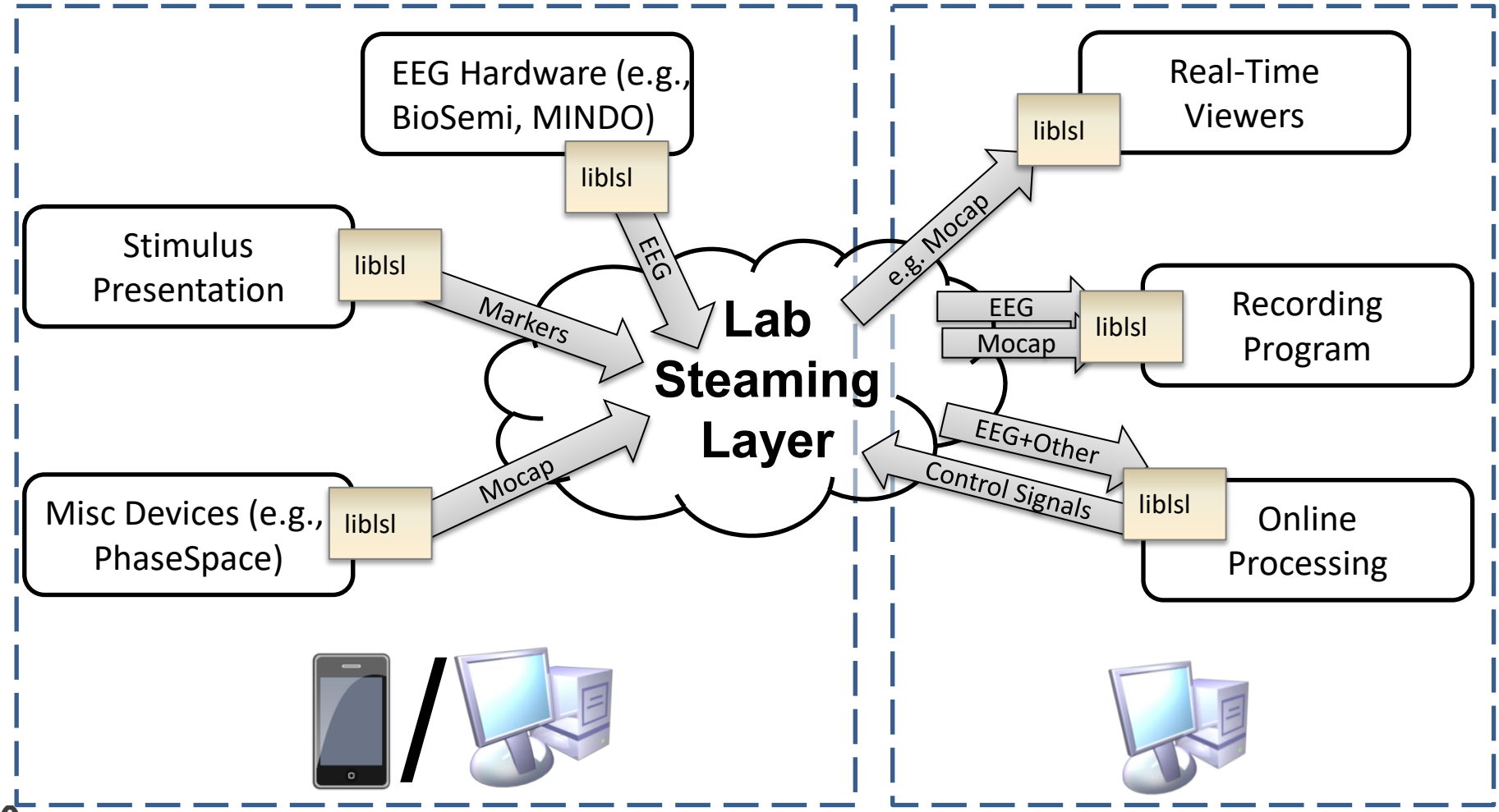
- Started in 2012 at SCCN
- Funding from ARL, ONR, NIH, DARPA, NASA
- Quite popular (32k demo views on YouTube)
- Goals
  - Unify real-time data access and recording file formats across many device types (EEG/EXG, NIRS, MoCap, Eye tracking, Audio, etc) and hardware/software vendors
  - Same workflow for both simple experiments and complex multi-modal / multi-person etc. setups
  - Provide built-in time-synchronization solution

Design Tradeoffs

- Designed for “lab-scale” recording operations:
  - Local: use VPN/broker/bridges to scale across the internet
  - Up to 20 streams per computer fine, 30-100 considered heavy load, likely needs high-end hardware beyond 100 streams (limited by # of USB ports, etc.)
  - Up to 10 computers involved per recording fine, >20 considered excessive, likely requires high-end networking equipment beyond 50 computers

- Designed for “human-scale” operating range:
  - Not a perfect fit for high-energy physics
  - Sub-milisecond time synchronization out of the box (microsecond precision can only be achieved with user-supplied (e.g., GPS/PTP) time stamps)
  - Latency <1ms, throughput up to 2MHz and 100MB/s (raw video)

The LSL Data Flow



LSL Can Be Easily Integrated Into Programs

```
% instantiate the library
lib = lsl_loadlib();

% make a new stream outlet (name: BioSemi, type: EEG, 8 channels, 100Hz)
info = lsl_streaminfo(lib,'BioSemi','EEG',8,100,'cf_float32','myuid');
outlet = lsl_outlet(info);

% send data into the outlet, sample by sample (8 random numbers each)
while true
    outlet.push_sample(randn(8,1));
    pause(0.01);
end
```

Sample code for sending 8ch EEG (MATLAB)

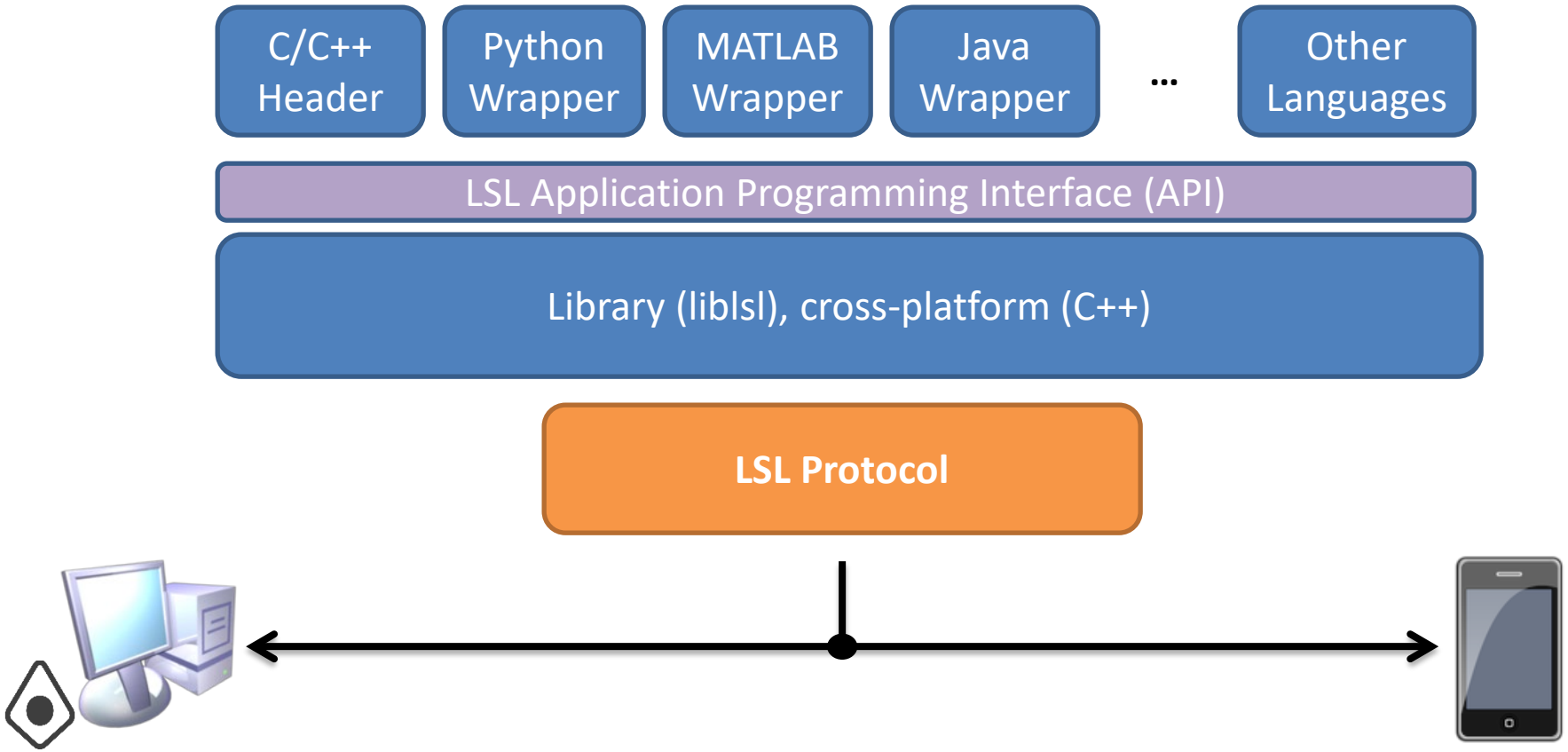
LSL Can Be Easily Integrated Into Programs

```
% instantiate the library
lib = lsl_loadlib();

% try resolve an EEG stream...
result = {};
while isempty(result)
    result = lsl_resolve_byprop(lib,'type','EEG'); end

% create a new inlet from the first result
inlet = lsl_inlet(result{1});

while true
    % get data from the inlet and print it
    [vec,ts] = inlet.pull_sample();
    fprintf('%.2f\t',vec); fprintf('%.5f\n',ts);
end
```

Sample code for receiving EEG data (MATLAB)

The LSL Software Stack

- The core piece of LSL is a network protocol, a library, and various language interfaces for it



The LSL Software Ecosystem

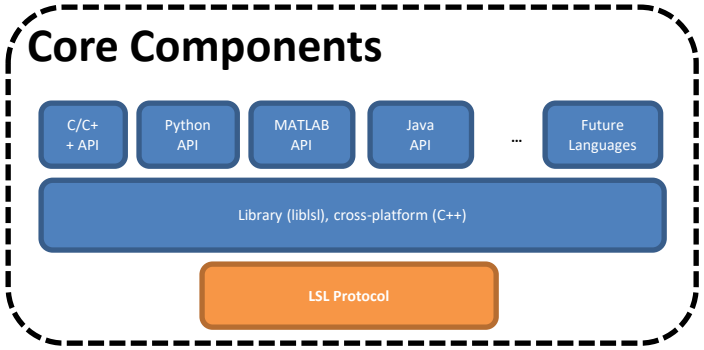
- The larger ecosystem includes Documentation, User Guides, Example Programs, Acquisition Programs, Generic Tools
- Largely open source (except vendor solutions)

Acquisition Programs (EEG, Eye tracking, Human Interfaces, Motion Capture, Multimedia)

Generic Viewers, Recorder

Example Programs

Wiki Documentation

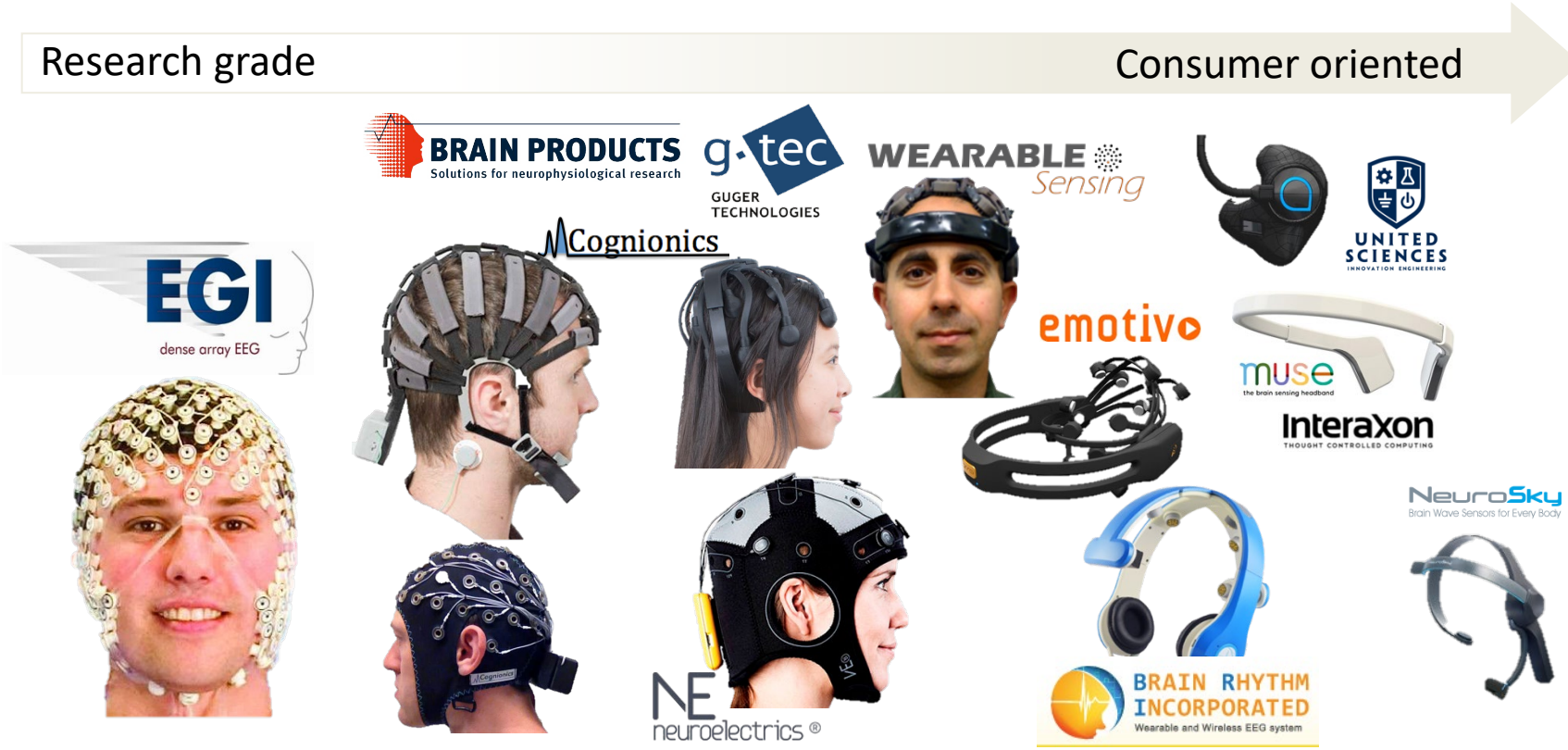


LSL Support from Industry

*Incomplete List

Some EEG Solutions Supported by LSL

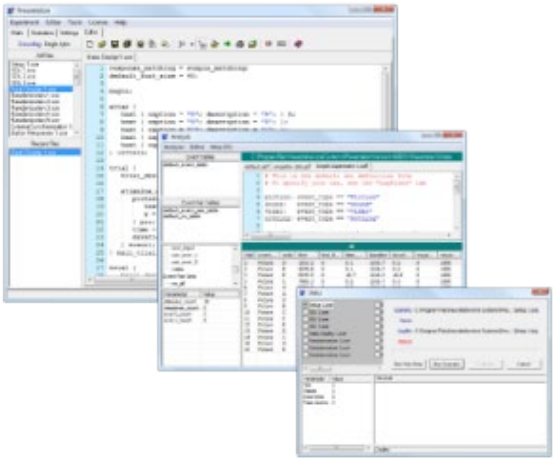
LSL supports 30+ EEG systems and over 20 other device classes



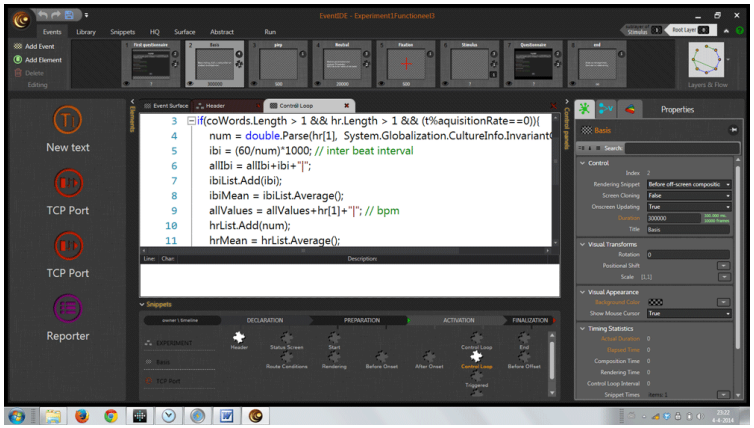
Some Other Device Types on LSL

- Eye Trackers
- Motion Capture
- Game Controllers
- Mice, Keyboards
- Serial Port
- Soundcards & (some) frame grabber cards
- Wearable EMG/ECG devices
- Photodiodes, temperature probes, etc

Some LSL-Compatible Stimulus Presentation Software



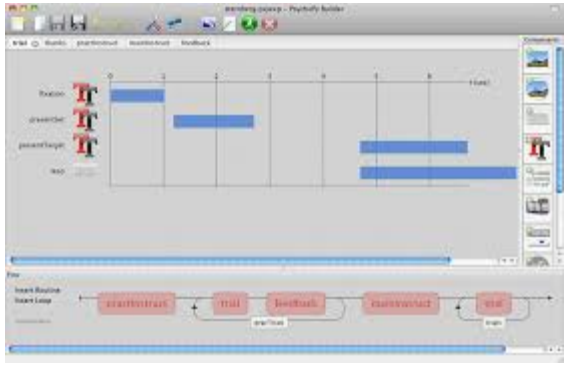
Presentation



EventIDE

unity

Unity (with plugin)



PsychoPy

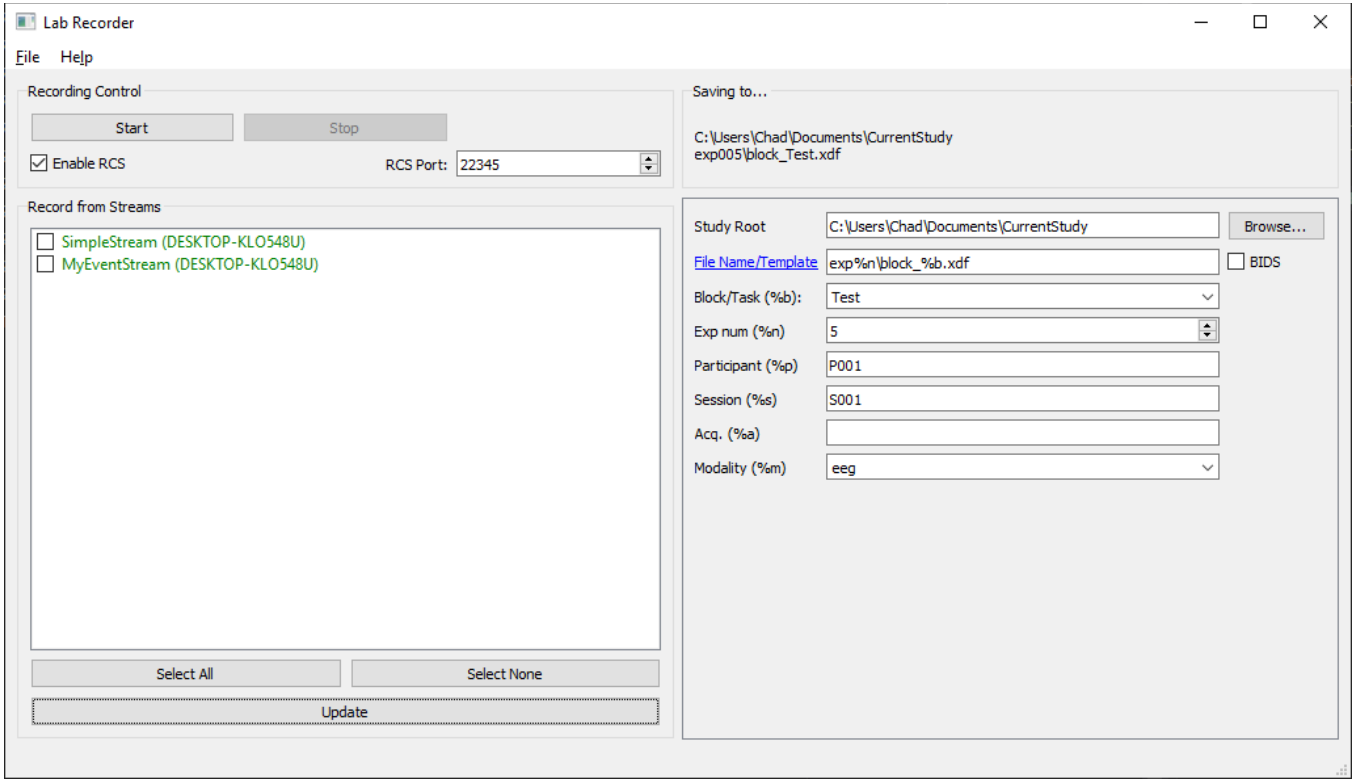
powered by Psychtoolbox

PsychToolbox

UNREAL ENGINE

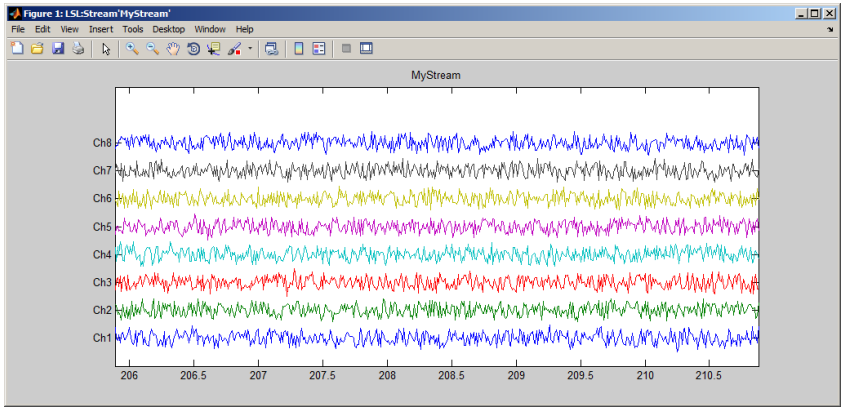
Unreal (with plugin)

Useful Tools: LabRecorder

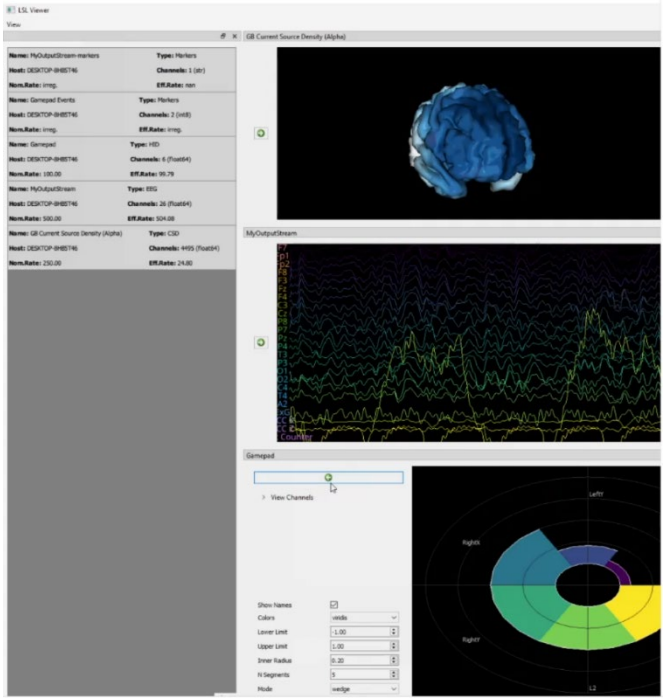


The LabRecorder can record any number of LSL streams simultaneously into a single file (XDF)

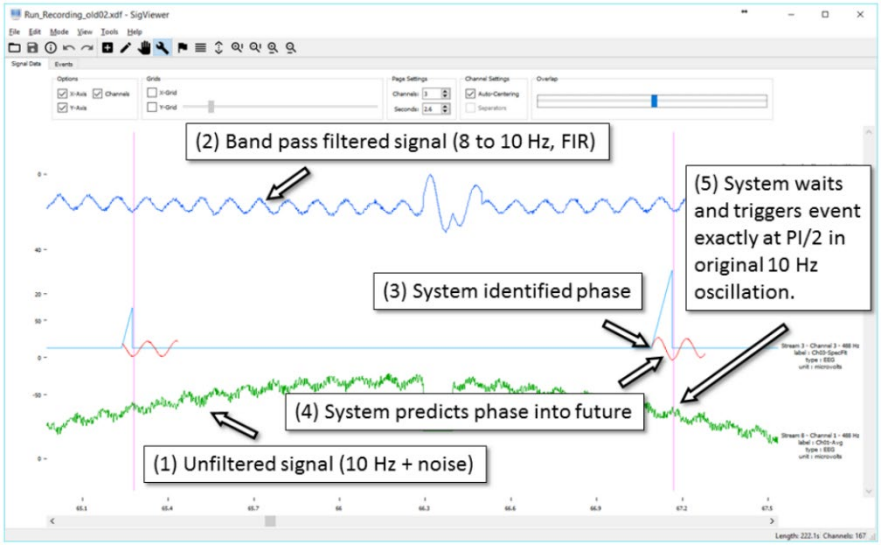
Useful Tools: Viewers



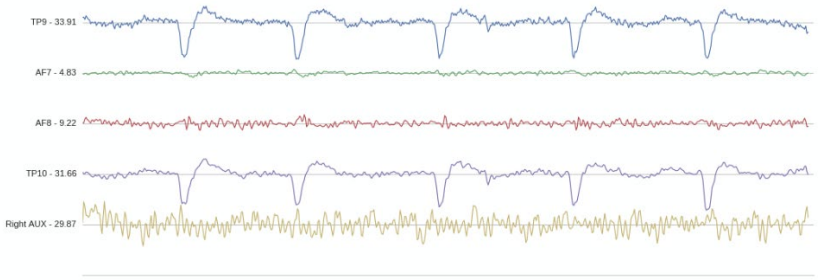
MATLAB Viewer (included)



LSL Viewer

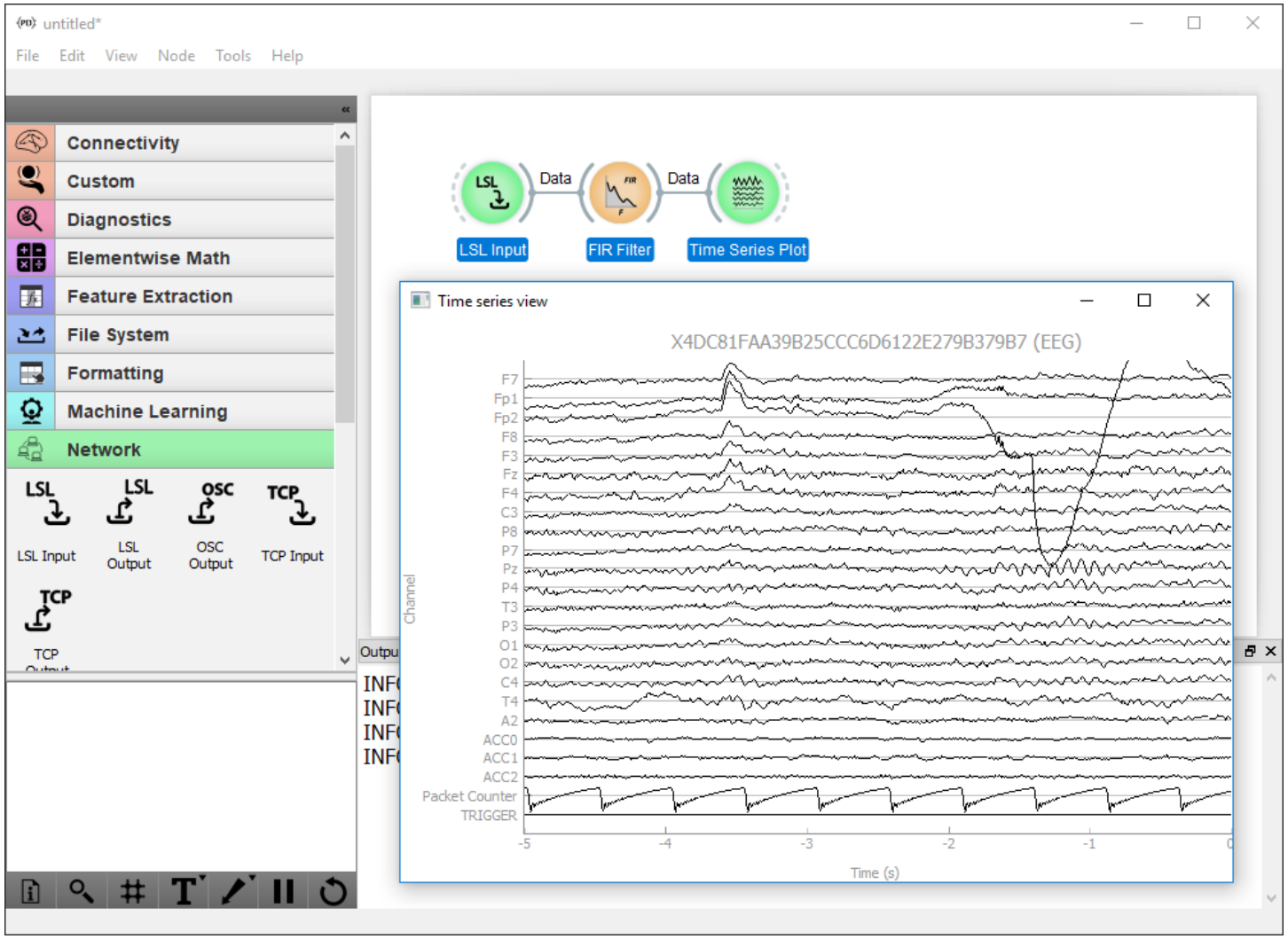


SigViewer (offline)



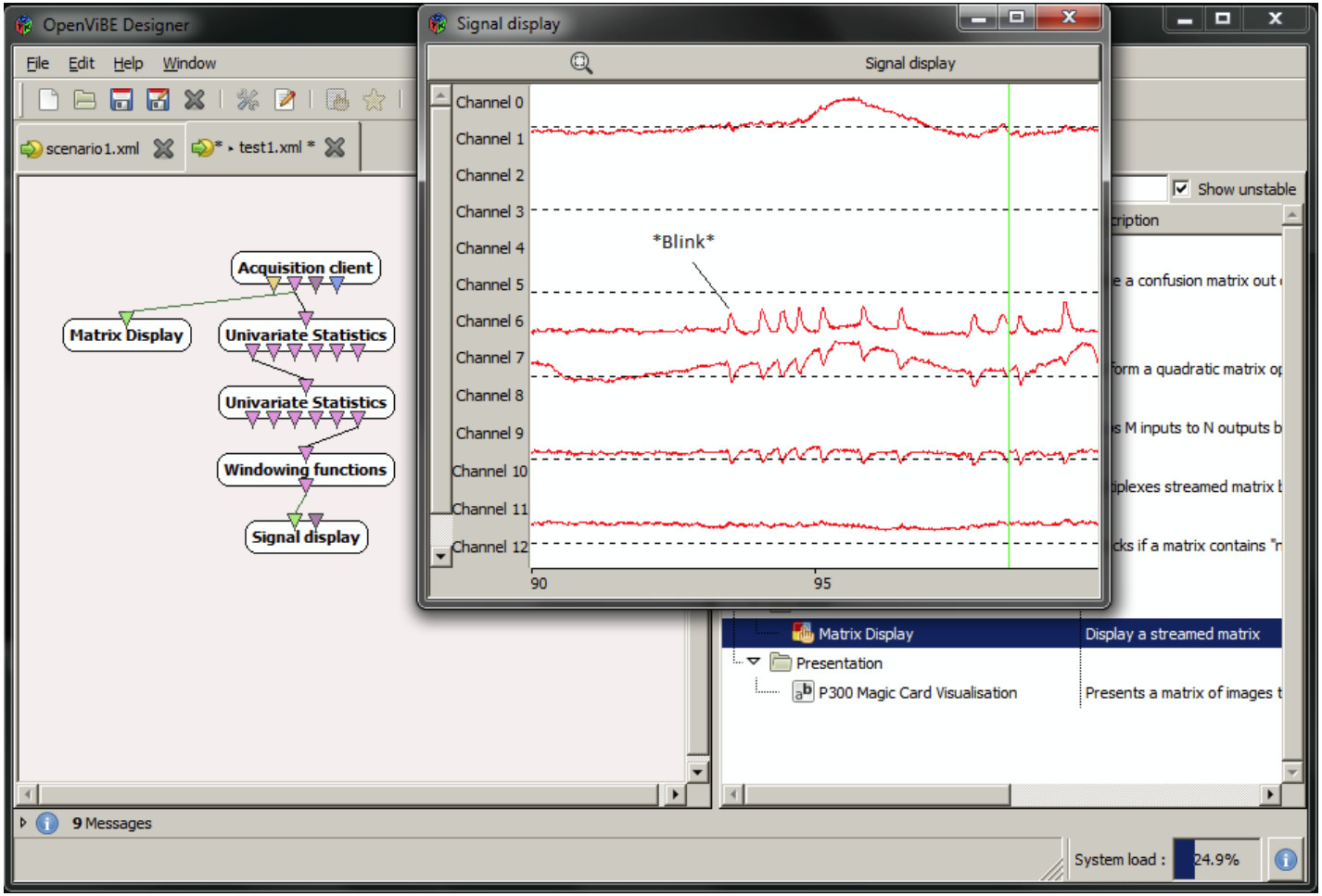
MuseLSL Viewer

Useful Tools: Real-Time Processing



NeuroPype Academic Edition

Useful Tools: Real-Time Processing



OpenViBE

Useful Tools: Command-Line Utils

- LSL comes with small utilities out of the box
- Can quickly diagnose network issues etc.
- E.g., `FindAllStreams`, `ReceiveData`, `SendData`, `ReceiveStringMarkers`, `SendStringMarkers`
- Generally available for all platforms

3 Using LSL

(Quick Demo)

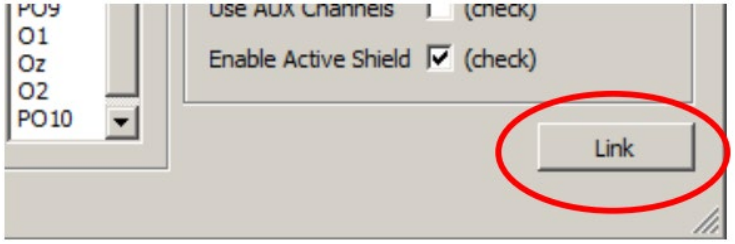
A Typical Experiment Setup with LSL

- “Record data from 2 devices while running a custom stimulus presentation script”

- Software needed for recording
  - Your experiment script (sends event markers)
  - Vendor A Application (e.g., sends EEG)
  - Vendor B Application (e.g., sends MoCap data)
  - Recording Program (LabRecorder)

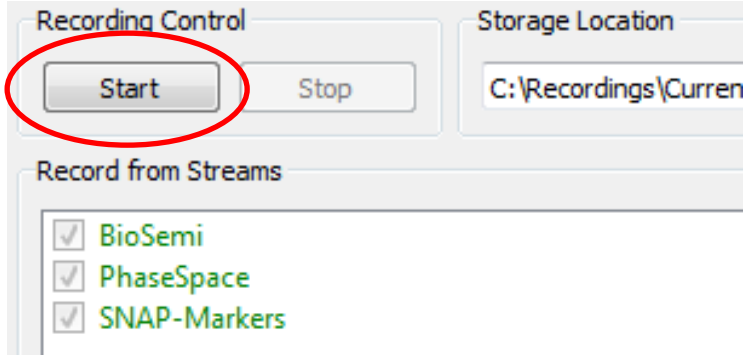
A Typical Experimenter Workflow

1. Start EEG & MoCap apps, turn on LSL streaming if needed



2. Start experiment script in ready mode

3. Open LabRecorder, confirm all LSL streams are there, and then click “Start”



Coding with LSL: Event Markers

```python
import random
import time

from pylsl import StreamInfo, StreamOutlet

# declare your marker stream information
info = StreamInfo('MyMarkerStream', 'Markers', 1, 0, 'string', ‘myuniqueid2345')

# create an outlet, now the stream is visible
outlet = StreamOutlet(info)

while True:
    # send an event marker
    outlet.push_sample(["Some Event Marker"])
    # do something else
    time.sleep(random.random()*3)
```

Example Code for sending event markers over LSL (Python)

Coding with LSL: Sending Time Series

```python
import time
from random import random as rand

from pylsl import StreamInfo, StreamOutlet

# create stream info
info = StreamInfo('BioSemi', 'EEG', 8, 100, 'float32', 'myuid34234')

# create an outlet
outlet = StreamOutlet(info)

while True:
    # make a new random 8-channel sample and send it
    mysample = [rand(), rand(), rand(), rand(), rand(), rand(), rand(), rand()]
    outlet.push_sample(mysample)
    # wait for a bit until we send the next sample
    time.sleep(0.01)
```

Example Code for sending a multi-channel time series over LSL (Python)

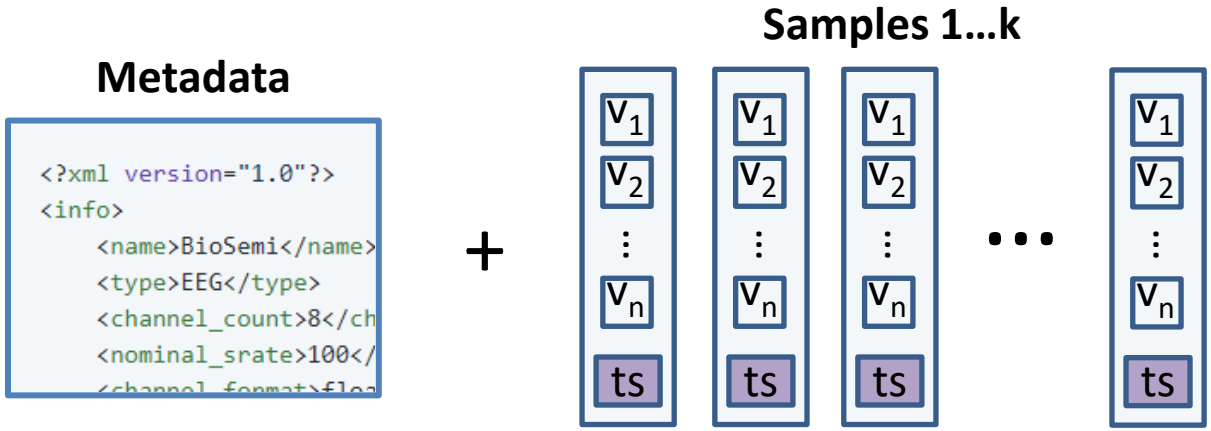
Coding with LSL: Receving Time Series

```python
from pylsl import StreamInlet, resolve_stream

# we wait until we find a stream with type EEG on the lab network... (or
more than one)
streams = resolve_stream('type', 'EEG')

# now that we have it, we create an inlet to read from it
inlet = StreamInlet(streams[0])

while True:
    # wait to get the next sample, also get its timestamp
    sample, timestamp = inlet.pull_sample()
    print(timestamp, sample)
```

Example Code for receiving a multi-channel time series over LSL (Python)

Some Facts Worth Knowing

- LSL doesn't reorder samples – the data you get out on the other side is always in-order
- LSL doesn't spuriously drop or lose samples (unless the network connection is interrupted for a long time, default 5 min.)
- For LSL, it's all just samples: one program can send whole chunks at a time, and the other side can read it sample-by-sample, or vice versa



- When a program first starts reading from a stream, it will begin reading from the stream's next submitted sample onward (e.g., from sample #10053 on)

Some Facts Worth Knowing

- You can add any amount of meta-data to a stream, and for posterity's sake, you *should:*

```
info = StreamInfo('BioSemi', 'EEG', 8, 100, 'float32', 'myuid2424')

# add some meta-data (follow the spec at https://github.com/sccn/xdf/wiki/Meta-Data)
info.desc().append_child("reference").append_child_value("label", "Nasion")

# add some more meta-data
channels = info.desc().append_child("channels")
for c in ["C3", "C4", "Cz", "FPz", "POz", "CPz", "O1", "O2"]:
    chan = channels.append_child("channel")
    chan.append_child_value("name", c)
    chan.append_child_value("unit", "microvolts")
    chan.append_child_value("type", "EEG")
```

- For best compatibility, LSL apps should adhere to the meta-data conventions set forth by the XDF (Extensible Data Format) project, which can be found at: https://github.com/sccn/xdf/wiki/Meta-Data

Thanks!

Questions?

Next speaker: Arnaud Delorme