BLUE BRAIN PROJECT

ZeroEQ

Zero Event Queue — Easy, safe, fast messaging for C++

What is ZeroEQ?

- C++ library for event-driven distributed ecosystems
  - ZeroMQ: robust and efficient data transport
  - Zeroconf: optional automatic resource discovery
  - ZeroBuf: Efficient, strong typed, strong semantic Serializables
- Push-based event propagation
  - Subscribe to events, receive them when they appear
  - Stateless, loose coupling to event publishers
- HTTP GET/PUT server with JSON payload

Example: Publish-Subscribe

Emitting application

```
zerobuf::render::Camera camera;

zeq::Publisher publisher;

while( true )
    publisher.publish( camera );
```

Receiving application

```
zerobuf::render::Camera camera;

zeq::Subscriber subscriber;

subscriber.subscribe( camera );
subscriber.receive();
```

- Zeroconf auto-subscribes within the same session (=user)
  - zeq::Subscriber(URI) for explicit subscription
- Receive blocks for one event

Pub-Sub Features

- Publisher
  - Publish objects with simple Serializable interface
  - ZeroBuf generates Serializable objects
    - to|from Binary|JSON
- Subscriber
  - Register Serializable objects
  - Updated upon receive()

```
servus::Serializable::setUpdatedFunction( const std::function< void() >& )
```

Example: HTTP Server

HTTP Server

```
zerobuf::render::Camera camera;

zeq::http::Server server( ”:4020” );
server.register_( camera );

while( true )
    server.receive();
```

Web Client

```
> telnet localhost 4020
Escape character is '^]'.
GET /zerobuf/render/Camera HTTP/1.1

HTTP/1.0 200 OK
Content-Length: 197

{
   "origin" : {
      "x" : 0,
      "y" : 0,
      "z" : 1
   },
   "lookAt" : {
[…]
```

HTTP Features

- Synchronous due to HTTP design
- Behaves like a combined publisher+subscriber
- GET for registered objects
- PUT for subscribed objects
- JSON payload

Use Cases

- Loosely couple similar applications
  - https://github.com/HBPVIS/Lexis: Camera, selection, time, transfer function, histogram, simulation results, …
- Bridge C++ applications into web services
  - Decouple GUI from application
  - http::Server + Javascript
- Remotely execute components of an application
  - Computation, IO, …
- Build distributed data processing applications

Gotchas

- Stateless
  - Slow subscribers will drop messages
  - Initial messages lost
- Passive push to active subscription
  - Publisher ignorant of subscribers by design
- Deaths and births of processes
- HTTP is synchronous request-reply

To Do

- Introspection for http::Server
  - GET /registry -> list of namespaces/objects, GET|PUT
  - GET /namespace/object/schema -> JSON schema of object
- Management server
  - Query schema, state and subscriptions
  - Initiate subscribe to a concrete publisher and event
  - For manual configuration through a “nodegraph” GUI
- Request-Reply support