TENOR 2015
INTERNATIONAL CONFERENCE ON TECHNOLOGIES FOR MUSIC NOTATION & REPRESENTATION

Introduction to INScore

D. Fober - *GRAME - Centre national de création musicale*

Inscore Workshop - May 28 2015

The Interlude Project

New Digital Paradigms for Exploration and Interaction of Expressive Movement with Music.

The Interlude Project

INScore

A framework for the design of augmented interactive music scores.

- An opened graphic and time space
- Time synchronization in the graphic space
- Performance representation
- Process activity representation
- Interaction
- A scriptable environment

INScore

INScore supports

- Symbolic music notation [GMN, MusicXML]
- Textual elements
- Bitmaps [jpg, gif, tiff, png,...]
- Vectorial graphics (rectangles, ellipses, SVG,...)
- Video files
- Sound and gesture graphic representations

INScore is

- a standalone score viewer
- an open source C/C++ library
- multi-platform
- an Open Sound Control API

DEMO

INScore

Relations between graphic and time space

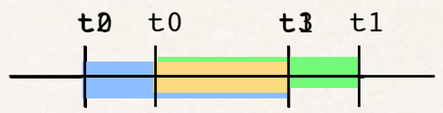
Hypothesis

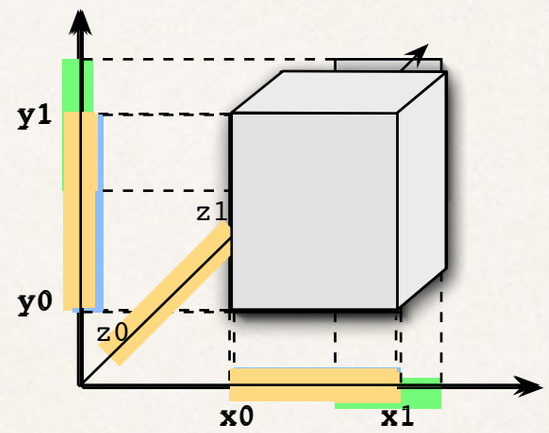
Approach the problem with segmentation and relations between segments

Segments

Defined as a list of intervals:

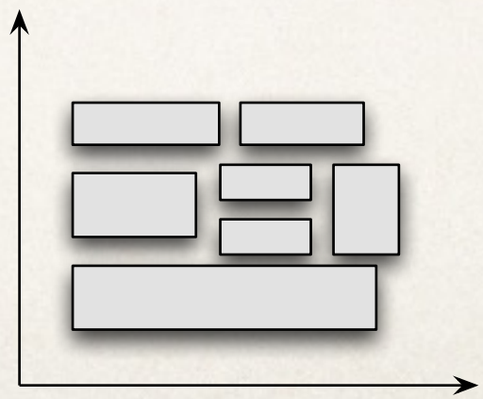
- property : empty (?)
- intersection operation
- generalizable to n dimensions





Segmentation

A set of disjoined segments



Relations between graphic and time space

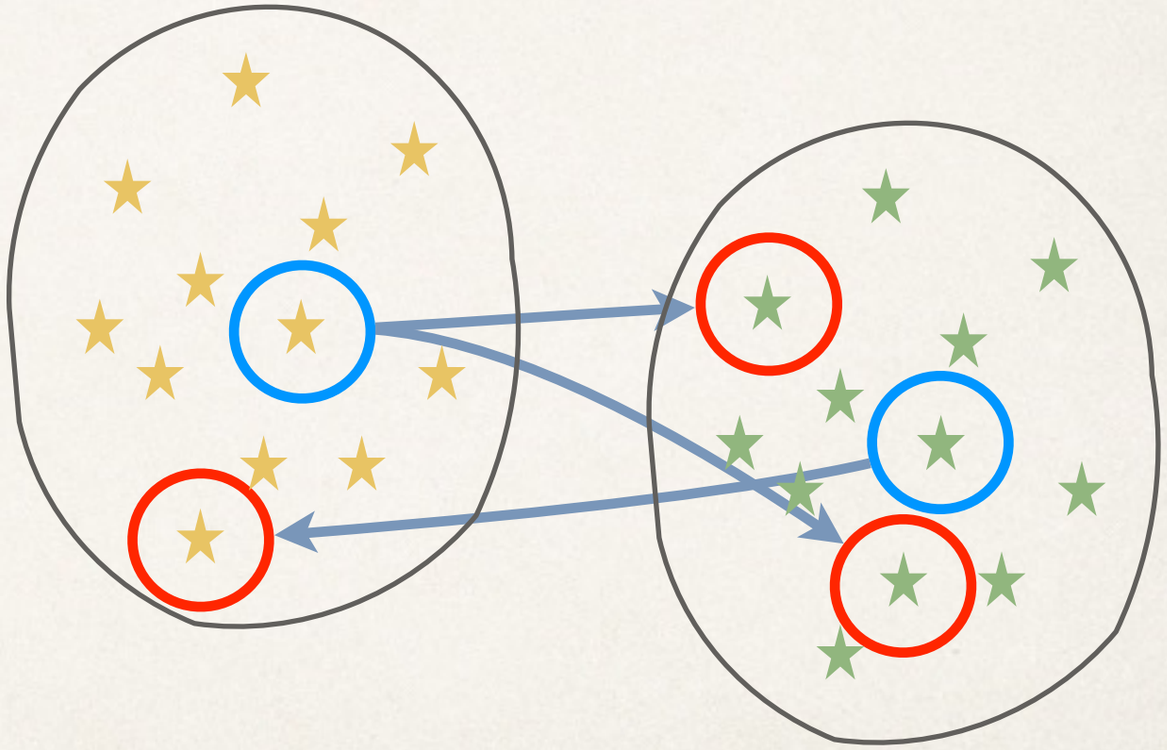
Hypothesis

Approach the problem with segmentation and relations between segments

Mapping

Relation between two segmentations:

- operations to query the mapping



Relations between graphic and time space

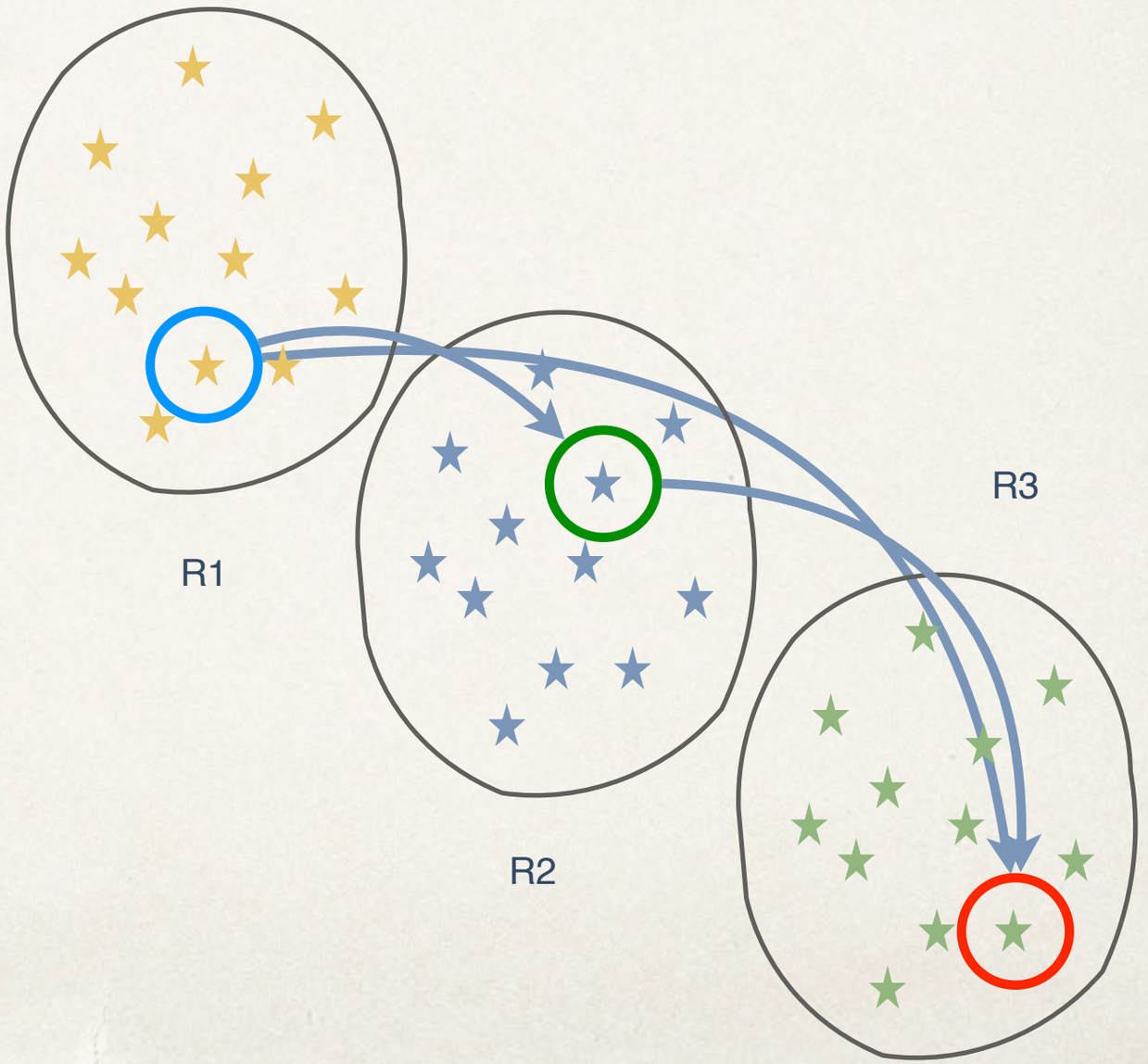
Hypothesis

Approach the problem with segmentation and relations between segments

Mapping

Relation between two segmentations:

- operations to query the mapping
- operations to compose mappings



Relations between graphic and time space

Segmentations and mappings for each component type:

type	segmentations and mappings required
text	*graphic* ↔ **text** ↔ **relative time**
score	*graphic*↔ *wrapped relative time* ↔ *relative time*
image	*graphic* ↔ **pixel** ↔ **relative time**
vect. graphics	**vectorial**↔ **relative time**
signal	*graphic* ↔ **frame** ↔ **relative time**

DEMO

INScore

Performance representation

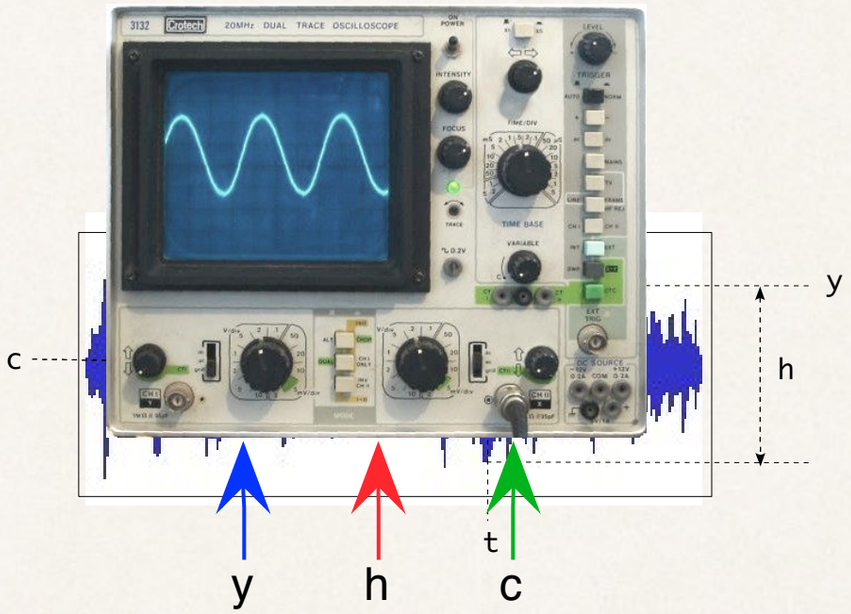
Hypothesis

Approach the graphic of a signal as a *graphic signal*.

A graphic signal

A composite signal made of:

- a *y* signal
- a thickness signal
- a color signal



Consider a signal *S* defined as a time function: $f(t) : \mathbb{R} \to \mathbb{R}^3 = (y, h, c) \mid y, h, c \in \mathbb{R}$

This signal could be directly drawn (i.e. without additional computation)

Performance representation

Signals parallelisation

Let $\mathbb{S}$, the set of signals $s : \mathbb{N} \to \mathbb{R}$.
We define a *parallel* operation '/' as:

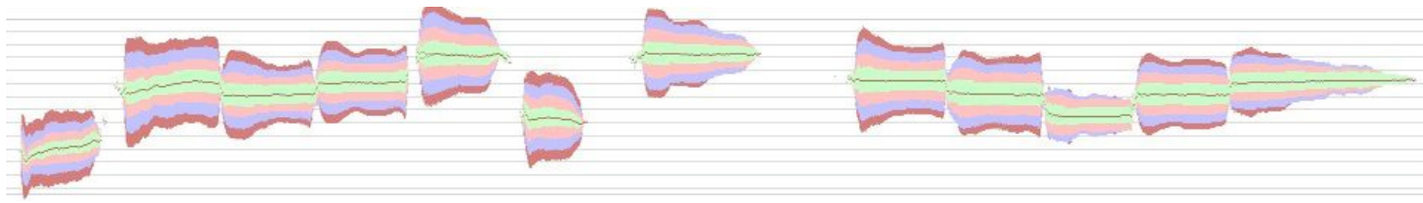
$$s_1/s_2/\ldots/s_n : \mathbb{S} \to \mathbb{S}^n \mid s_i \in \mathbb{S}$$

Time function of a parallel signal $s^n \in \mathbb{S}^n : \mathbb{N} \to \mathbb{R}^n$

$$f(t) = (f_0(t), f_1(t), \ldots f_n(t)) \mid f_i(t) : \mathbb{N} \to \mathbb{R}$$

Performance representation
System expressivity

Examples



$g0 = S_{f0} \ / \ S_{rms0} \ / \ k_c0$

S_{f0} : fundamental frequency

S_{rms0} : f0 RMS values

$g1 = S_{f0} \ / \ S_{rms1} + S_{rms0} \ / \ k_c1$

S_{rms1} : f1 RMS values

$g2 = S_{f0} / \ S_{rms2} + S_{rms1} + S_{rms0} \ / \ k_c2$

S_{rms2} : f2 RMS values

...

$g = g2 \ / \ g1 \ / \ g0$

DEMO

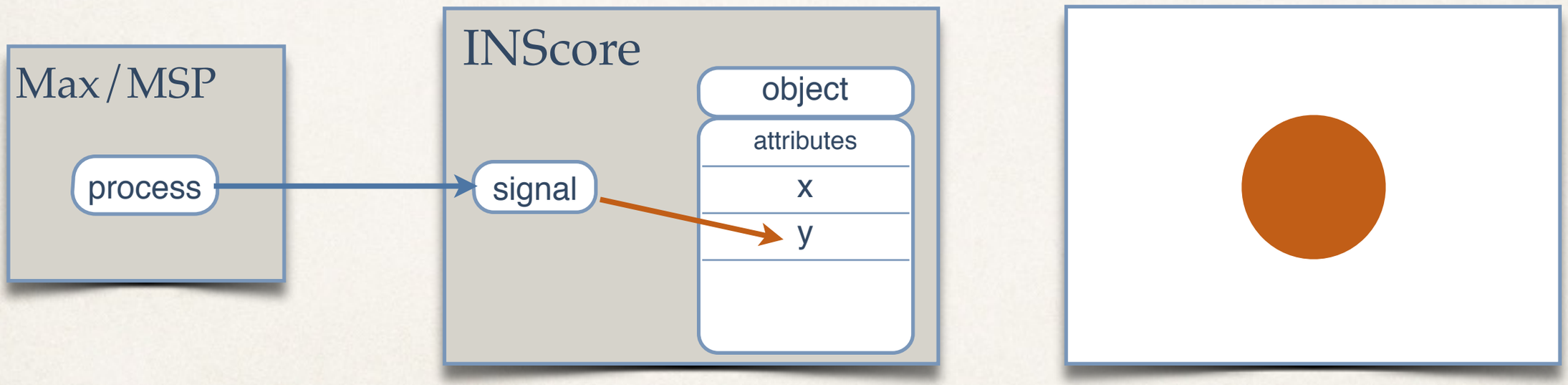
INScore

Process activity representation

Hypothesis

A process activity may be viewed as a signal.

Connecting signals to graphic attributes



DEMO

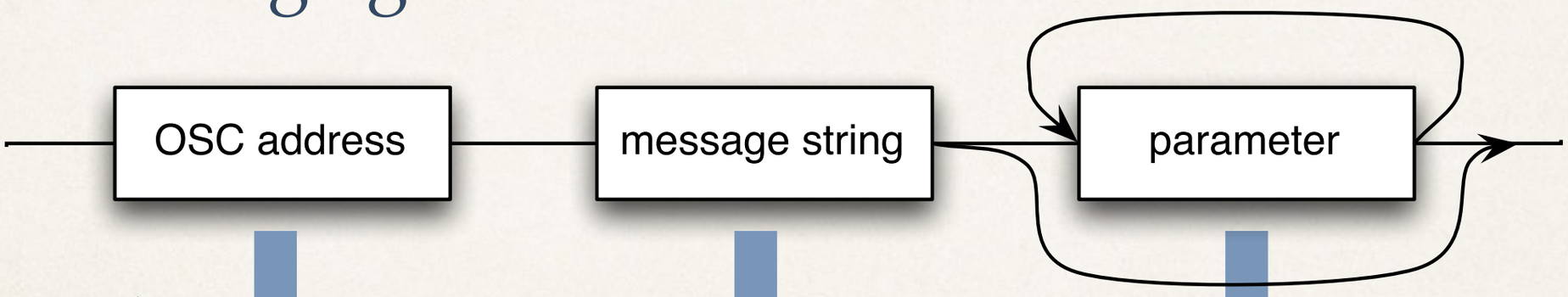
INScore

INScore OSC Messages

An object oriented approach

- The OSC address is like an object pointer.
- An OSC message is similar to an object method call.
- The OSC address space is dynamic.

OSC message general format

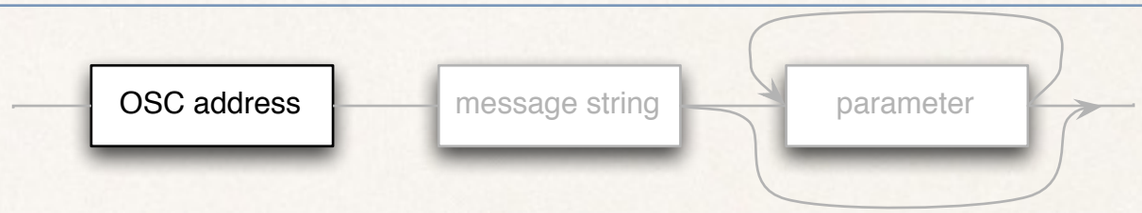


Example

```
/ITL/scene/score  color   255 128 40 150
```

```
score->color(255, 128, 40, 150)
```

INScore OSC Address Space



Application

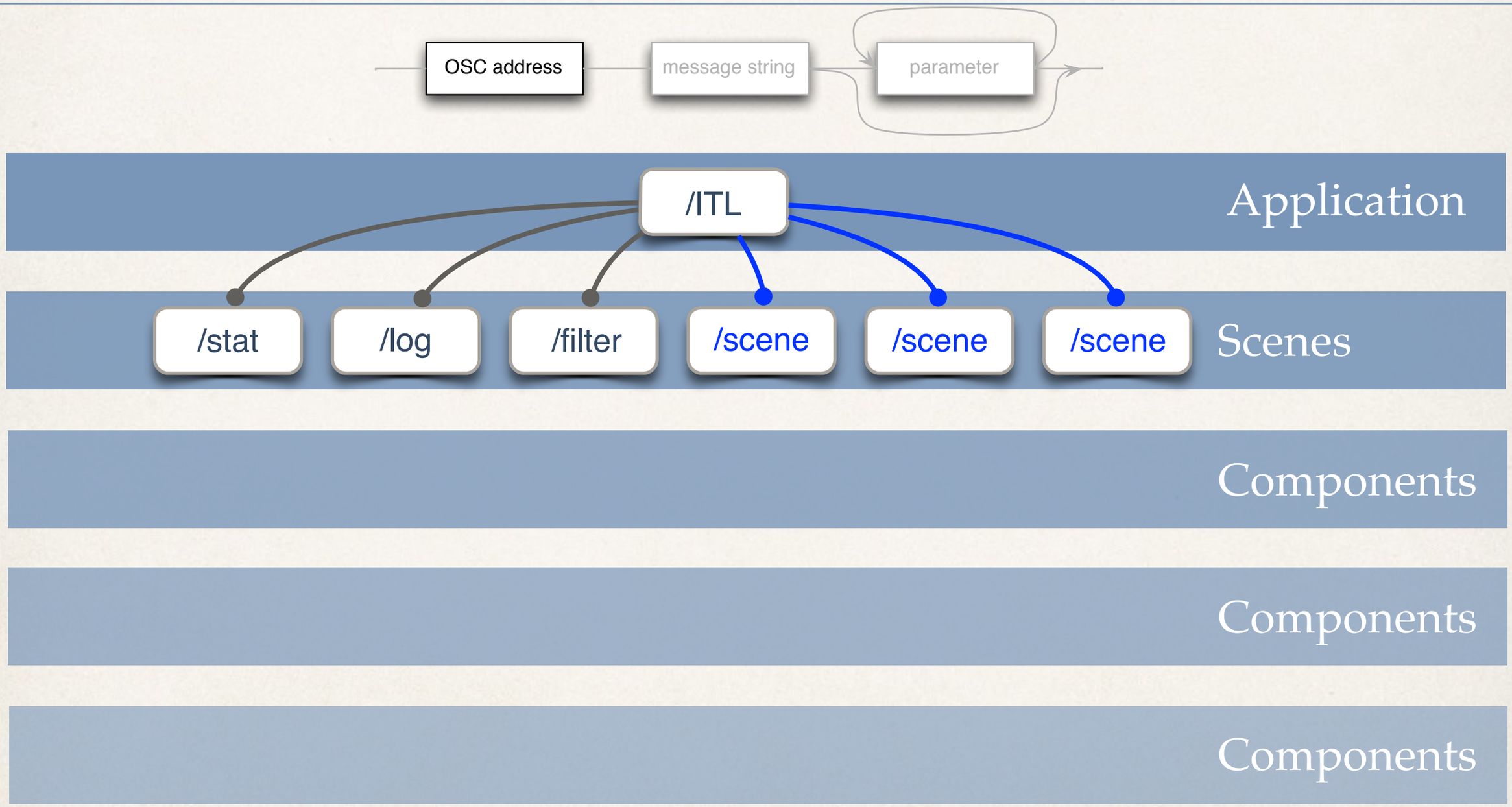
Scenes

Components

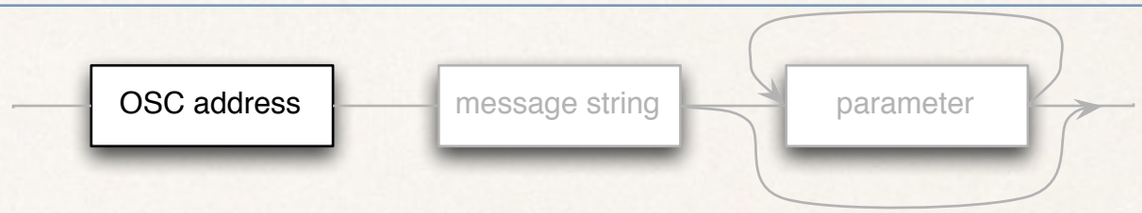
Components

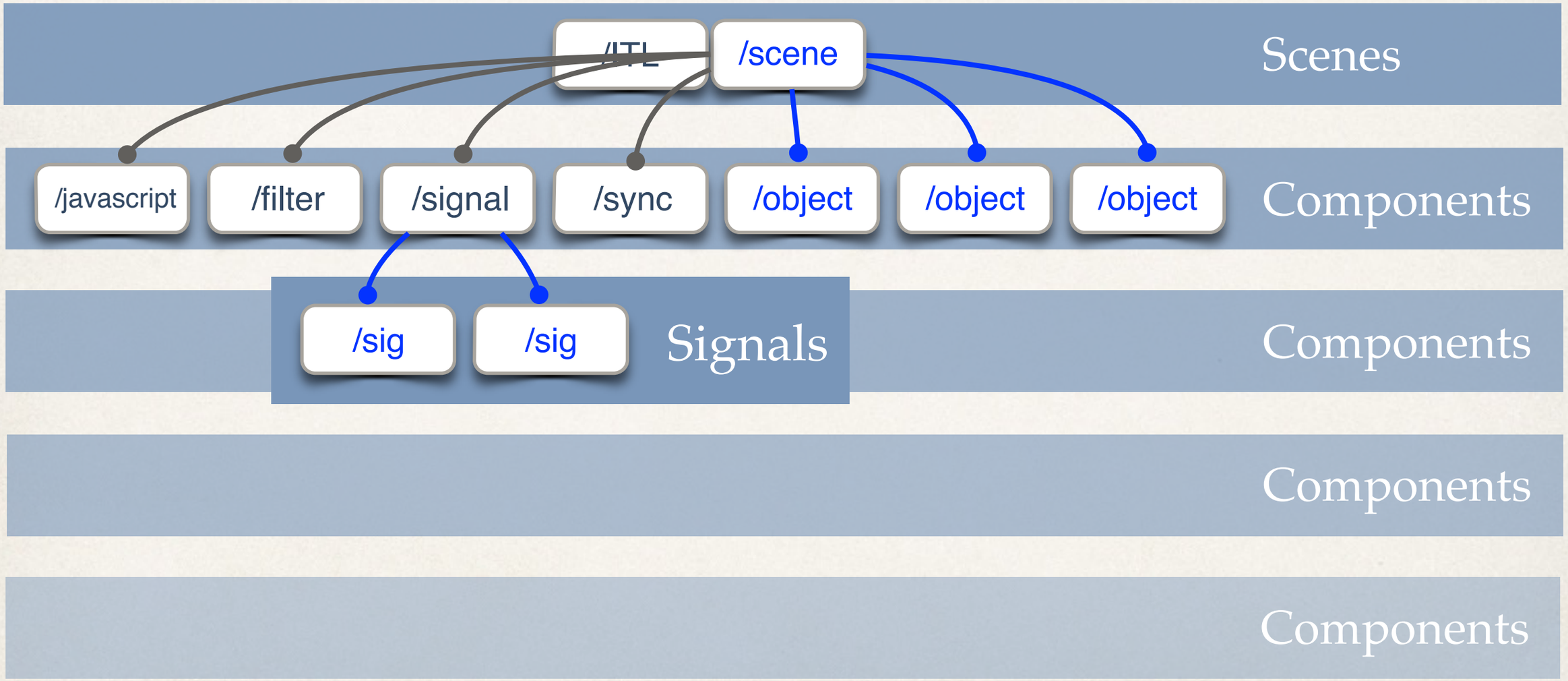
Components

INScore OSC Address Space

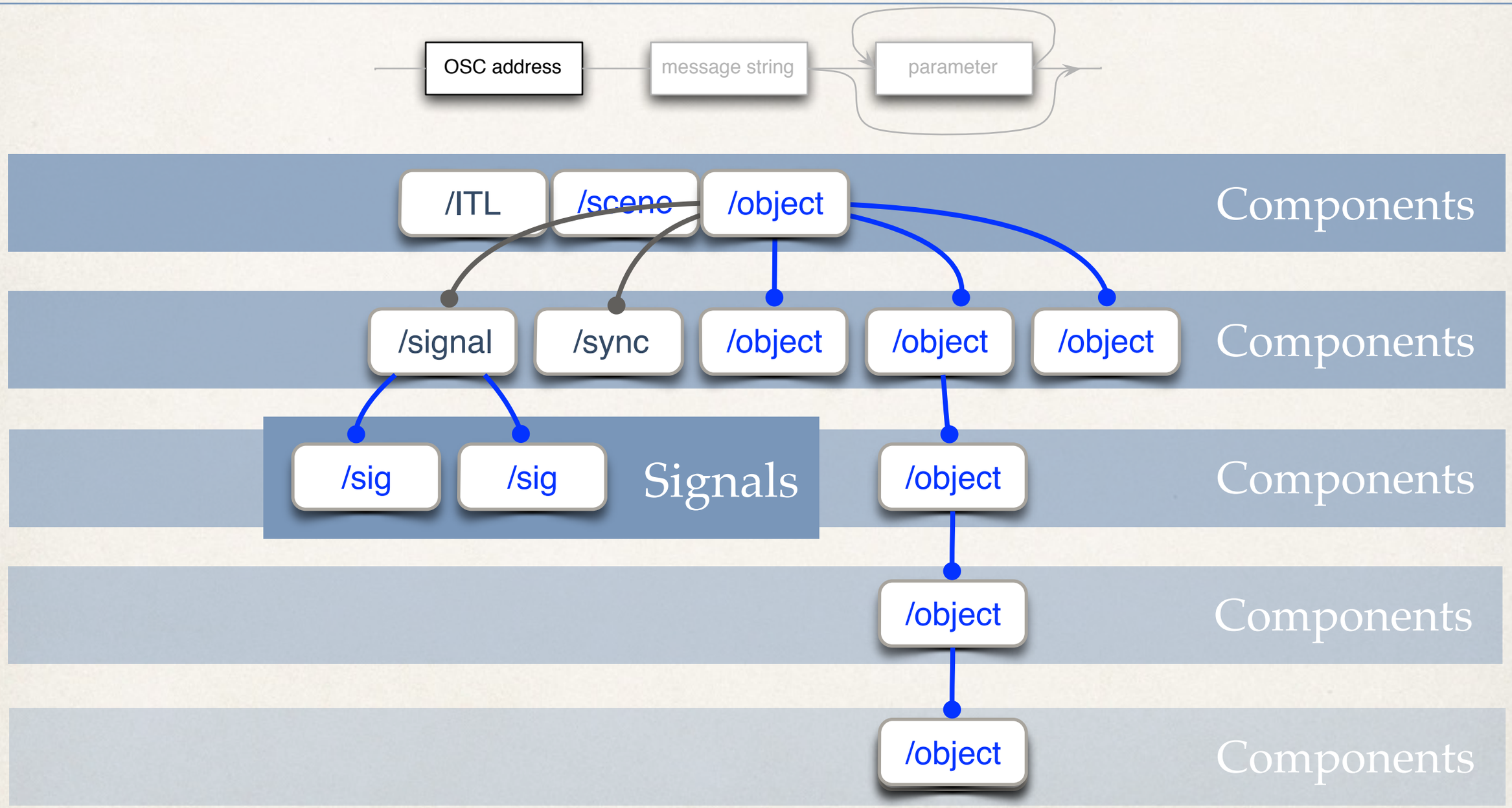


INScore OSC Address Space

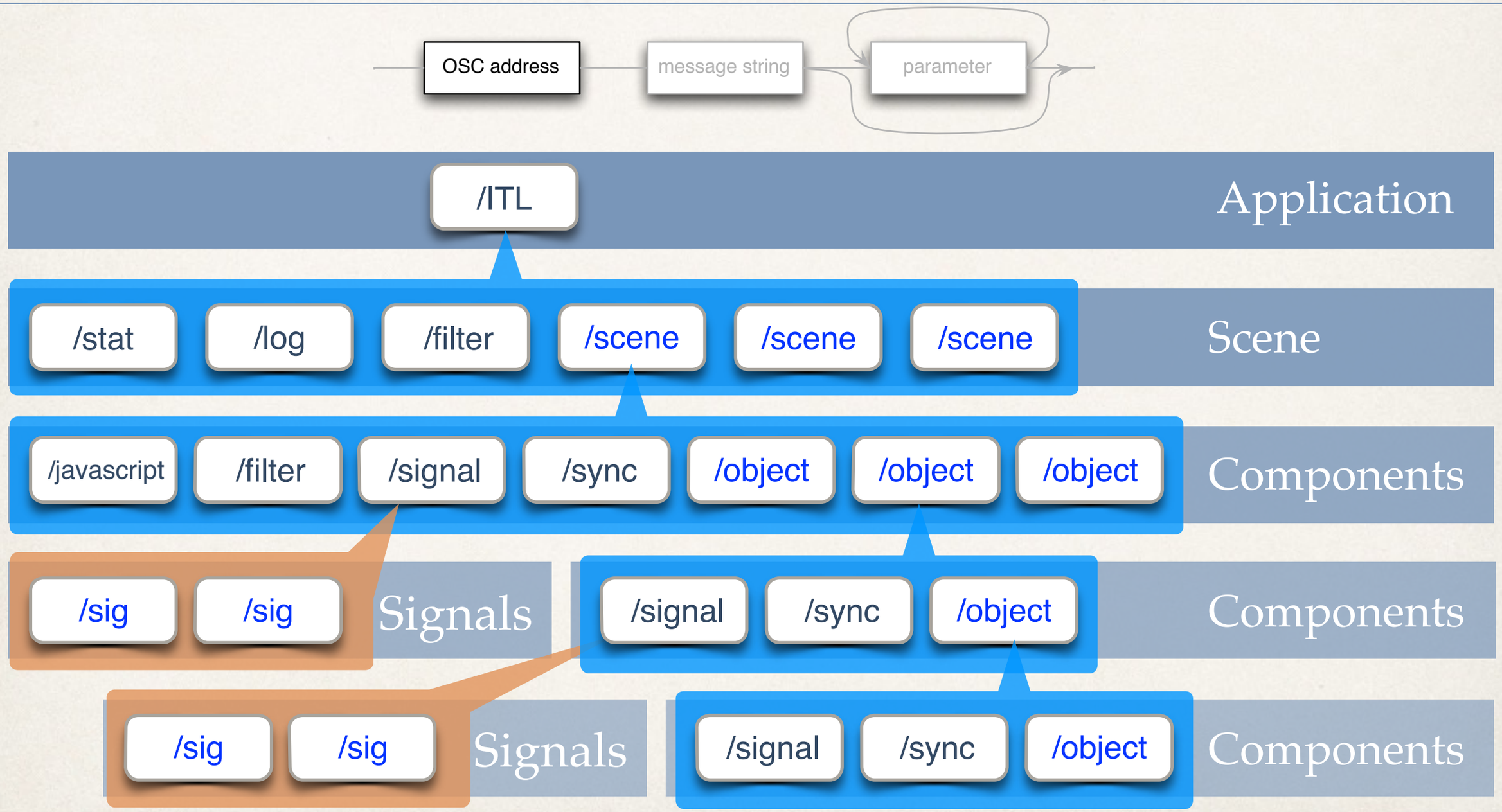




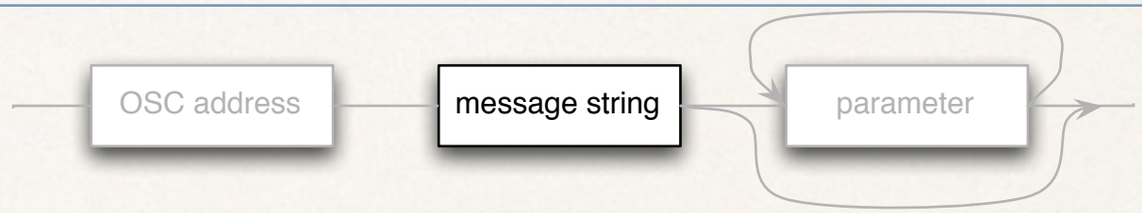
INScore OSC Address Space



INScore OSC Address Space



Messages Strings



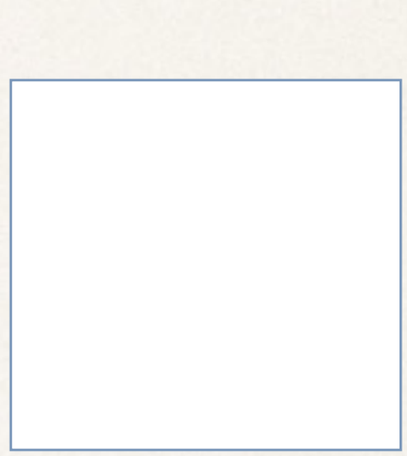
Graphic space control

- Position:

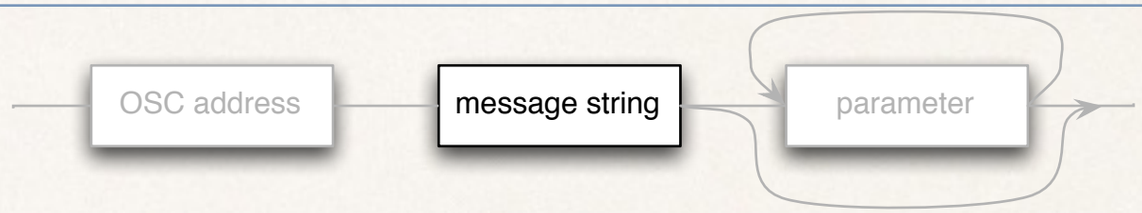
  (d)x, (d)y, (d)z, (d)scale,(d)angle
  (d)xorigin, (d)yorigin,
  (d)rotatex, (d)rotatey, (d)rotatez

- Color:

  (d)color, (d)red, (d)green, (d)blue,
  (d)hsb, (d)hue, (d)saturation,
  (d)brightness,
  (d)alpha



Messages Strings



Time space control

- Time position:
  (d)`date, clock,`

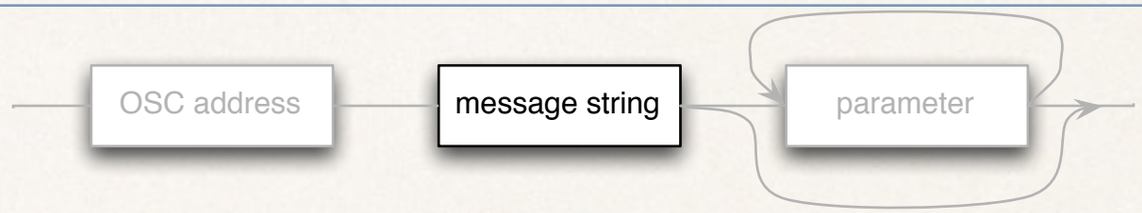
- Duration:
  (d)`duration, durclock`

Date forms

- Rational expressed as:
  n d
  n (1)
  float value
  “n/d”

1 =

Messages Strings



Constructor

- `set <type> args`

Textual components

-
-

Symbolic music notation

-
-
-

Images & video

-
-
-

Vectorial graphics

- `svg(f)`
- `rect`
- `ellipse`
- `…`

Signals

-
-

Web objects

-
-

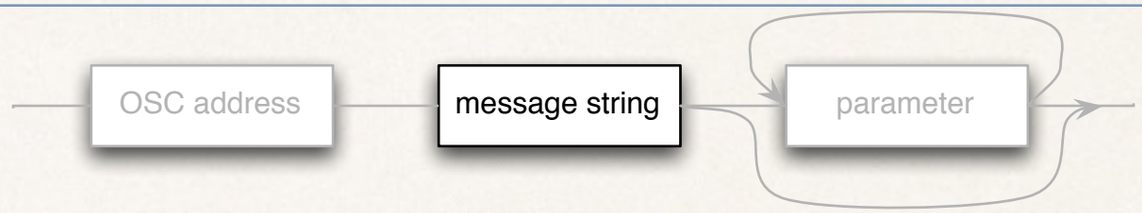
Plugins

-
-
-

Misc

- `layer`
- `grid`
- `file`

Messages Strings



Queries

- `get`
- `get <attributes>`

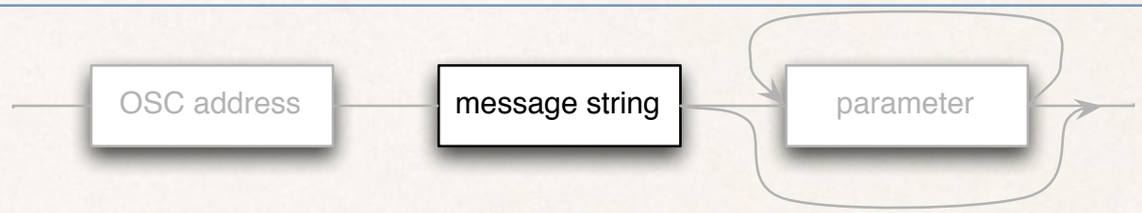
```
/ITL/scene/object get
```

```
/ITL/scene/object set type <params>
```

```
/ITL/scene/object get x  y scale
```

```
/ITL/scene/object x  0
/ITL/scene/object y 0.3
/ITL/scene/object scale 1.2
```

Messages Strings



Mapping

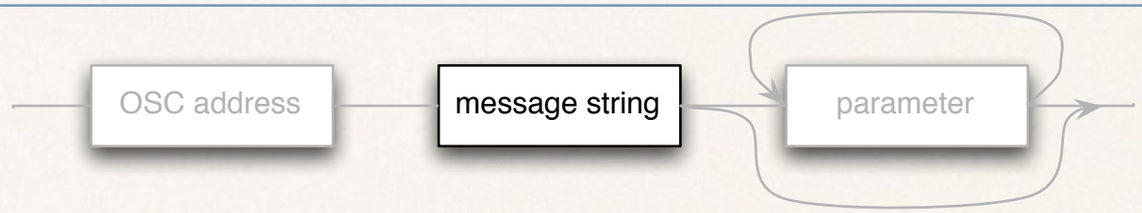
- `map(f)`

Graphic interval

Time interval

```
/ITL/scene/object map "( [0, 901[ [15, 191[ )  ( [0/4, 4/4[ )
                       ( [2, 772[ [223, 380[ ) ( [4/4, 8/4[ )";
```

Messages Strings



Misc

- `export`
- `save`
- `alias`
- `eval`
- `show`
- `del`

Application level

- `hello`
- `load`
- `rootPath`
- `forward`
- …

Javascript engine

- `run`

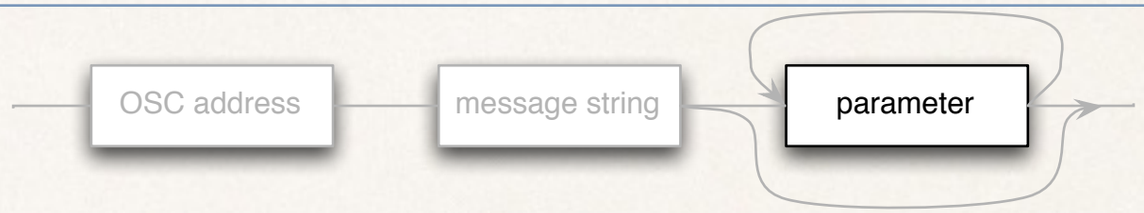
Signals

- `connect`
- `disconnect`

Scene level

- `new`
- `del`
- `load`
- `rootPath`
- `fullscreen`
- …

Messages Parameters



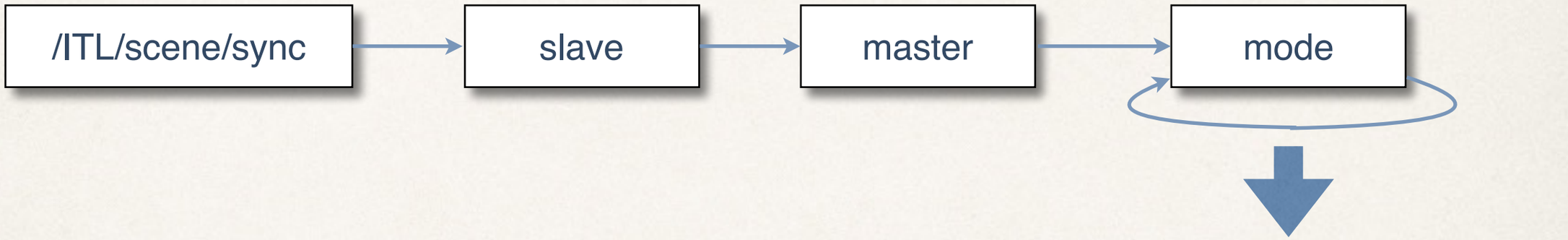
Direct use of basic OSC types

- `int32`
- `float32`
- `OSC-string`

Relaxed types

Strict parameters count

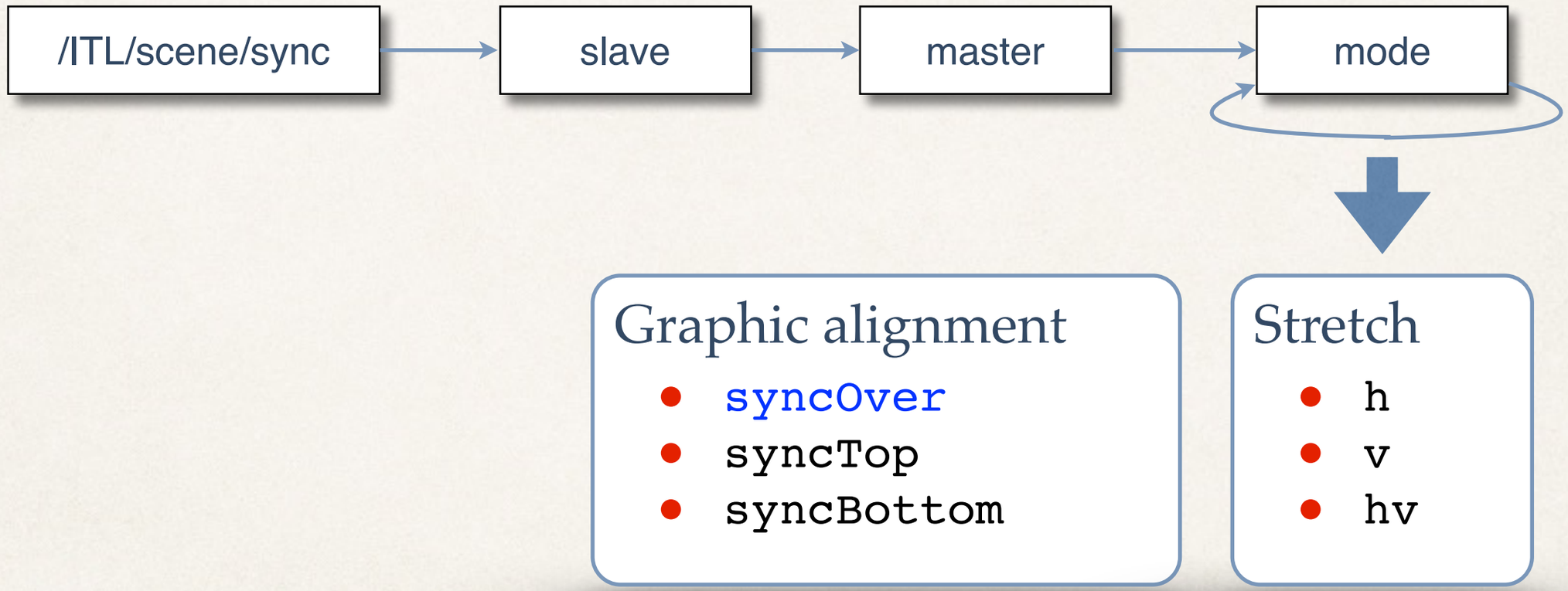
Synchronization



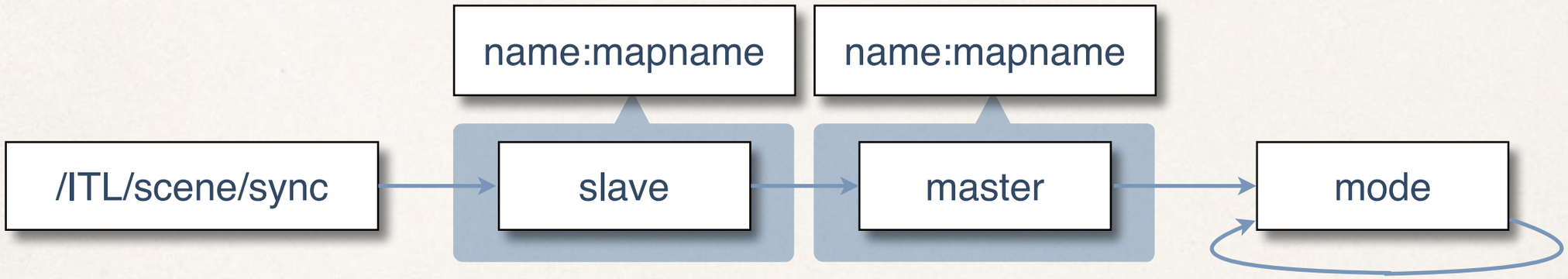
Graphic alignment

- `syncOver`
- `syncTop`
- `syncBottom`

Synchronization



Synchronization



Graphic alignment

- `syncOver`
- `syncTop`
- `syncBottom`

Stretch

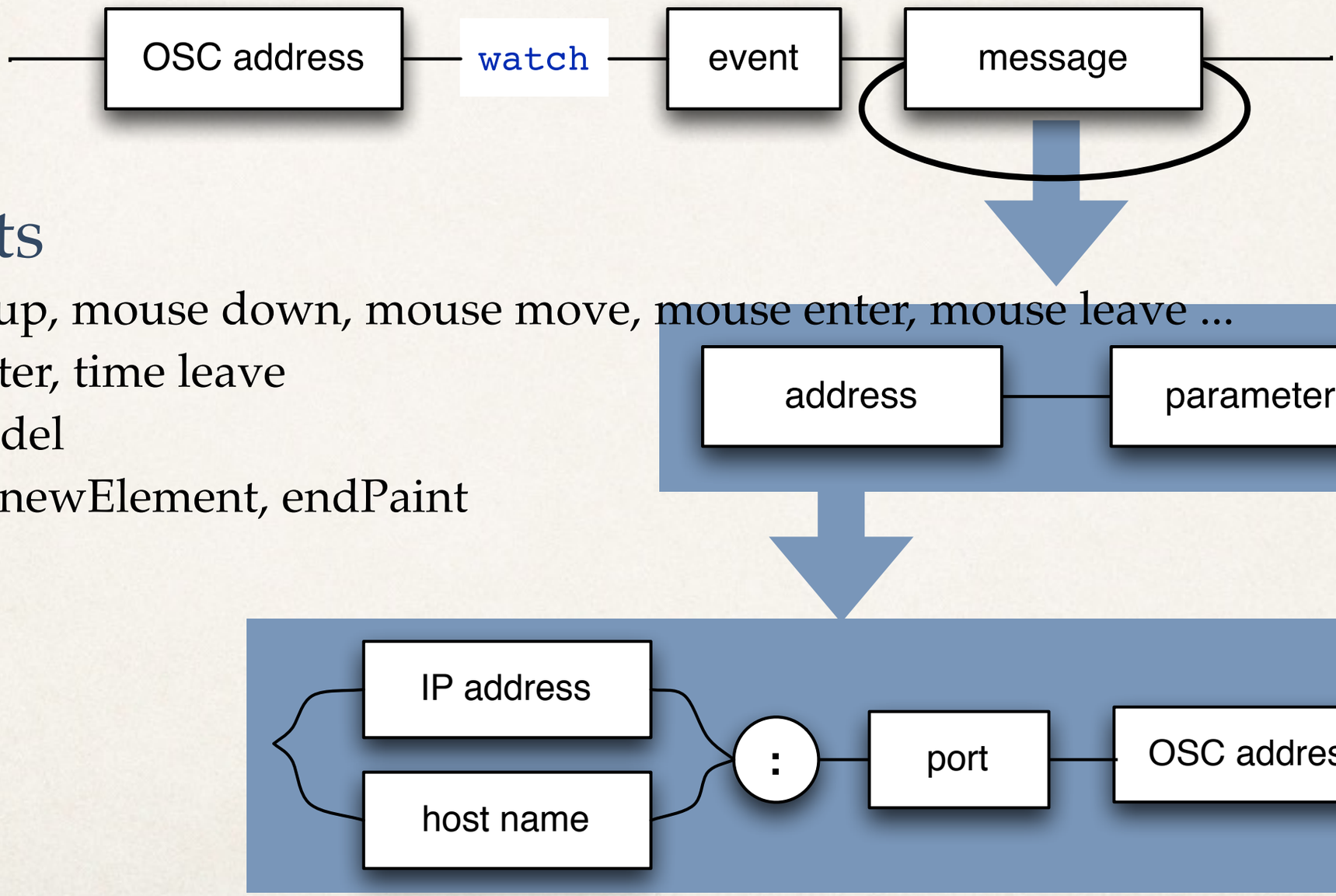
- `h`
- `v`
- `hv`

Time alignment

- `relative`
- `absolute`

Interaction Messages

Basic principle

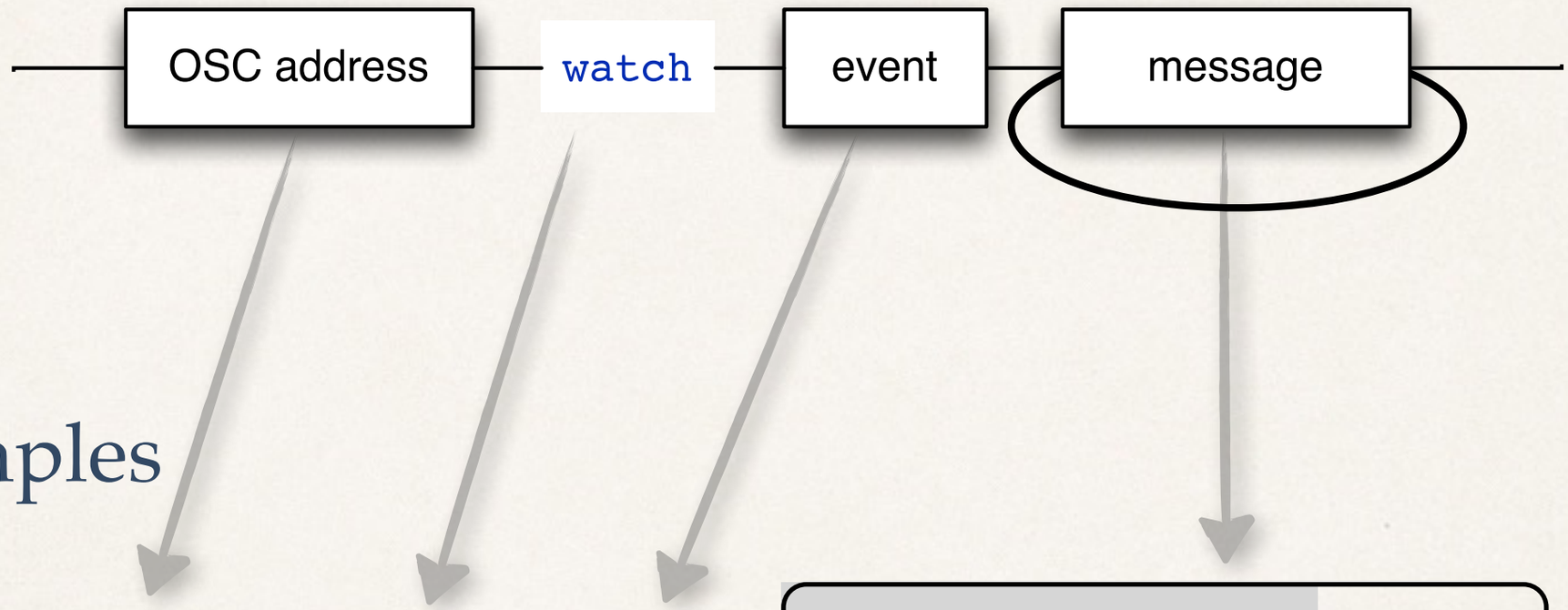


Events

- mouse up, mouse down, mouse move, mouse enter, mouse leave ...
- time enter, time leave
- export, del
- [scene] newElement, endPaint

Interaction Messages

Basic principle



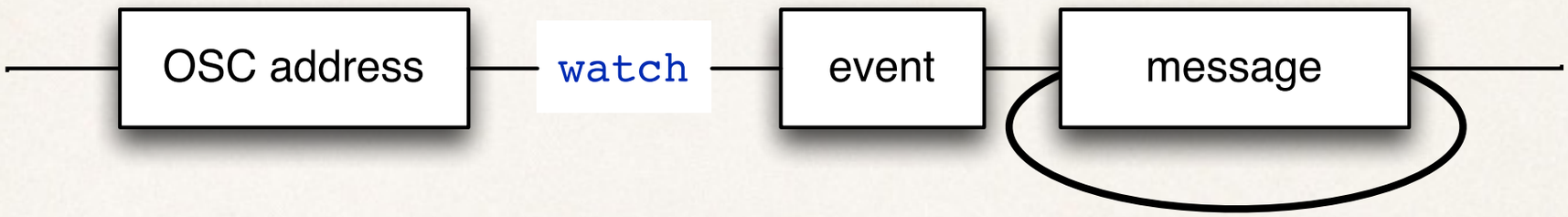
Examples

```
/ITL/scene/myObject watch mouseDown /ITL/scene/myObject show 0
```

```
/ITL/scene/myObject watch mouseDown localhost:8000/an/address start
```

Interaction Messages

Message as first class parameter




```
/ITL/scene/myObject watch event task1 then
                    watch event task2 then
                    watch event task3 then
                    watch event task4 then
                    ...
```

Interaction Messages

Interaction state management

- watch
- push
- pop
- event <what> <params>

An infinite loop

```
/ITL/scene/myObject watch event (push, task1) then
                    watch event task2 then
                    watch event task3 then
                    watch event (task4, pop)
```

Interaction Messages

Variables

- `$x, $y, $absx, $absy, $sx, $sy`
- `$date, $rdate`

Quantizing dates

- `$date[n/d]`

Message based variables

- `$(a valid INScore ‘get’ message)`

Scaling values

- `$x[min, max], $y[min, max]`

Address variables

- `$self`
- `$scene`

DEMO

INScore

INScore scripts

A textual version of OSC messages extended with:

- variables
- extended OSC addresses
- relative addresses
- message based parameters
- javascript sections

```
gray = 120;
color= $gray $gray $gray;
localhost:8000/vol 120;
msg = (./rect set rect 1 1, /ITL/scene/obj set color $color;
/ITL/scene/o1 x/rect color $gray);
      $(/ITL/scene/o2 get x);
/ITL/scene/obj eval $msg;
<?javascript
/ITL/log write o2 x is
   // javascript code
   $(/ITL/scene/o2 get x);
   // producing INScore messages
   // as output
?>
```

Example

Flux Aeterna

Flux Æterna has been composed by Vincent Carinola in 2014. The piece has been designed for the Internet. It comes under the form of an endless audio stream. The listening conditions are similar to those of a web radio but here, the listener can influence the future of the piece by providing its own sound files.

`http://vr.carinola.free.fr/fluxaeterna/`

Questions ?

http://inscore.sourceforge.net