EventFunc / EventDef

An EventFunc represents an articulable structure with memory that can turn on and off. It can contain functions, synths and task that start/stop and receive data all through a single interface. in the context of CtrlBuss they can also react to control signals that articulate how the interface reacts to incoming data.

EventFunc is a structure that can contain user customizable features that are controlled and articulated as parts of the whole. By default, EventFunc have two functions that coorespond with .start and .stop messages and a local environment. Additionally, synths and tasks can be added using .addSynth and .addTask methods. Interpretation of start, stop and task functions and arguments to synths that are made as functions are resolved in the local environment. By using the .set method, the EventFunc will update the environment and all dependant features like tasks, synths and nodes within the optional (?) local group node.

//
// needs more explanation
//
mantains \startTime, \lastTime, \state, calculates \interOnset

in the contect of CtrlBuss can be set to .canFocus(true) to receive signals

//
//
//

EventDef maintains the registry of EventFunc and provides an easily accessible interface

EventDef(name, startFunc, stopFunc);

name - a symbol
startFunc - a Function that will execute when .start method is performed. The first argument into the function is self
stopFunc - a Function that will execute when .start method is performed. The first argument into the function is self

This creates a new EventFunc of name if name does not already exist in the registry.

instance methods

.at - .at(key)
access the value in the local Environment associated with the symbol key

.put - .put(key, value)
associate the object value with the symbol key in the local Environment

.set(key, value, key2, value2, ..., keyN, valueN)
distribute values to associated parameters stored in the local Environment andas arguments to nodes.
The environment is updated with these values and synths and groups are set.
The parameter values in the environment are used to inform any synths and grains whose args contain it.
These parameter values are also available in startFunc, stopFunc and all added Task features

environment is updated and nodes are set with these values and

.updateAll(\key1, \value1, \key2, \value2, ..., \keyN, \valueN)
set key value pairs to the environment and nodes bypassing the disconnect structure.
Originally a private system method but may have some limited/special case uses for users.

.setTarget(target, addAction)
set the default target for items within the EventFunc
target - a node or the synbol for an existing GroupDef
addAction - \addToHead, \addToTail, \addBefore, \addAfter

.run(boolean)
pause and resume the synth and task processes.

.addBuffer(name, frames, channels)
a convenience method for creating a BufferDef. It registers the name of the buffers automatically passes [\buf, bufnum] to all dependents

.canFocus(bool)
communicate up the parent tree that this node and branch can register with the CtrlBuss focus scheme

.listen(bool)
set to false, disconnect \all for group 1 is enacted. All data from MapBuss are not updated on receipt.
set to true, all data from MapBuss are updated to the Environment and nodes.

.disconnectMapsLevel(type, group)
sets the disconnect state handling the receipt of data from MapBussfor the EventFunc
type is a symbol: \none, \all, \synth, \group, \allNodes

.disconnectParam(\param)
param - symbol of an argument used in a dependent to stop updating (seperate from disconnectMapsLevel)

.connectParam(\param)
param - resume updating the param

.setDisconnectGroup(int)
int - assign a number disconnect group to the EventFunc. The EventFunc will only respond to incomming disconnects for this group.
Disconnect groups 0 and 1 are system groups and are not normally set by the user.
However, group 0 can be set to remove an EventFunc from a group.

.addPatchMap(\param, \replacement)
both arguments are symbols.

replace a parameter with another as data is processed at input. the original parameter is not updated.

.removePatchMap(\param)
param is a symbol
remove a parameter patch. The incoming parameter will resume updating the EventFunc as input

.setWait(taskName, waitTime)
taskName - a symbol, the name of the task
waitTime - a number, the amount of time between executions of the task function

.setLoop(taskName, loopNum)
taskName - a symbol, the name of the task
loopNum = a number, set the number of iterations for the task

.addTask(name, taskFunc, stopFunc ...options)
name - a symbol, name of the task
taskFunc - a function, executed at every step of the task. the function is passed self and the step number.
stopFunc - a function, executes after the last execution of taskFunc. the function is passed self and the step number.
...options, aditional/optional key values pairs for \loopNum and/or \wait

```
EventDef(\eventName).addTask(\taskName, { |ed, ii| }, { |ed, ii| }, \wait, 0.1, \loonum, 4);
```

default values are \wait, 0.1, \loopnum, inf

.startTasks
start all tasks in the taskList. Commonly used internally and may only be of limited user application.

.startTask(taskName)
taskName - a symbol, the name of the task

start a task named taskName in the taskList. Commonly used internally and may only be of limited user application.

.pauseTasks

pause all tasks in the taskList. Commonly used internally and may only be of limited user application.

.resumeTasks

resume all tasks in the TaskList. Commonly used internally and may only be of limited user application.

.resumeTask(taskName)

resume taskName from the TaskList. Commonly used internally and may only be of limited user application.

.startLoop(taskFunc, ...options)
taskFunc - a function, executed at every step of the task. the function is passed self and the step number.
options are, in either order, a function and a number. The function is stopFunc. The number is the wait time between steps.

this method is useful for creating and starting a task from within an EventFunc starFunc. The method is another entry point for .addTask and works in private name space within the EventFunc. Repetative calls to .startLoop will stop, then overwrite existing tasks created with this method.

startFunc and stopFunc are evaluated in the local Environment.

```
EventDef(\test, { |ed|
	ed.startLoop(
		{ |self, ii| [self.name, ii].postln },
		{ |self, ii| [self.name, ii, \stopped].postln },
		0.125
	)
},{});

EventDef(\test).start;
EventDef(\test).stop;
```

.addSynth(synthName, ...args)
synthName - as symbol, name to store the synth
args - key value pairs that will be sent to the synth at .start

.addSynth stores the recipe for creating a client side synth object in the EventFuncs synthList.

values can be functions. The function will be evaluated in the local Environment and the resulting value will be sent to the synth. Functions will be evaluated on each .start.

It is important to note, that for each parameter that should be informed by the environment (and by extension, in a CtrlBuss scenario, data streaming from the MapBuss) that the key and a default value should be made in this args list. Otherwise, the parameter will remain unresolved and resort to the SynthDef default.

in the background, \gate, 1 will be sent with all start messages and at stop, .set(\gate, 0) will be sent so you do not need to manage gates manually.

in the background, if the environment contains a symbol value at \buf, the EventFunc will lookup that value in the BufferDef reistry and pass the buffer.bufnum as a key value [\buf, bufnum] to the synth.

.removeSynth(synthName)
synthName - as symbol, name to store the synth

removes a synth from the EventFuncs synthList

.startSynths
start all synths in the synthList. Commonly used internally and may only be of limited user application.

.startSynth(synthName)
synthName - a symbol, the name of the task

start a synth named synthName from the synthList. Commonly used internally and may only be of limited user application.

stopSynths
stops all running synths managed by the EventFunc. Commonly used internally and may only be of limited user application.

stopSynth(synthName)
synthName - as symbol, name to store the synth

stops a synth. Commonly used internally and may only be of limited user application.

.addGrain(grainName, ...args)
grainName - a symbol, the name of the grain recipe used to generate grains from within the EventFunc
args - parameter key value pairs

addGrain differs from .addSynth only in that the EventFunc does not maintain a connection to the grain.

Grain uses the the Synth.grain feature to create light weight synths without client side representation. These synths are forgotten as soon as they are launched so be sure to design the associated synthdef with that in mind. No .set(\gate, 0) will be sent to these synths.

see .addSynth for further discussion

.removeGrain(grainName)
grainName - a symbol, the name of the grain

remove a grain recipe from the EventFun grainList

.grain(grainName)
grainName - a symbol, the name of the grain

used to launch a grain built from the recipe stored at grainName in tge grainList of an EventFunc. Useful for launching grains from within startFunc, stopFunc and tasks.

.toggleState

.addToggle(startToggleFunc, stopToggleFunc)

toggle funcs are triggered by start. Functions are evaluated in the local Environment.Toggles are resolved before start/stop funcs, synths and tasks

.start

starts the EventFunc and all features (startFunc, toggle, synths, tasks)

the execution order of features is:

startToggleFunc/stopToggleFunc
startFunc
startSynths
startTasks

.stop

starts the EventFunc and all features (stopFunc, toggle, synths, tasks)

the execution order of features is:

stopFunc
stopTasks
stopSynths