GraphDef / GraphFunc

GraphDef maintains the registry of GraphFunc and provides a global interface to all nodes.

GraphDef allows the manufacture of SynthDef(s) based on a common wrapper.

GraphDef(name, func, ...args)
name - a symbol
func - a function
...args - symbols in pairs

GraphDef(\default, func, \type, \wrapper) //build a wrapper named default

GraphDef(name, func, \wrapper, \default) //build a graph that is wrapped in default

GraphDef(name, func) //build a plain old synthdef

Instance methods

.add(libname, completionMsg, keepDef) - add the synthdef to the SynthDescLib servers.
see SynthDef.add

.start(...args)
start a Synthdef with the initial control values args. the client side representation is stored in the environment.at(\synth)

.start(args, target, addAction)
start a Synthdef with the initial control values args at target and addAction. the client side representation is stored in the environment.at(\synth)

args - Optional Array of pairs of control name and values.
target - a server or node
addAction - \addToHead, \addToTail, \addAfter, \addBefore, \addReplace

.stop
if the GraphDef contains the control \gate, then a .set(\gate, 0) is send to the node. Otherwise, perform .free

//simple example

s.reboot;

GraphDef(\pink, { |outBus = 0| var sig; sig = PinkNoise.ar([0.1,0.1]); Out.ar(outBus, sig) });

GraphDef(\pink).start

GraphDef(\pink).stop

```
//a wrapper example

s.reboot;

//build the graph container
GraphDef(\default, { |outBus, gate| var sig;
	/*sig = _GRAPHDEF_; */
	sig = sig * EnvGen.ar(Env.adsr(0.1,0.1,1,0.25), gate, doneAction:2);
	Out.ar(outBus, sig);
}, \type, \wrapper);

//build the graph contents
GraphDef(\test, { |freq, amp| SinOsc.ar(freq, 0, amp); }, \wrapper, \default);

// test the GraphDef

GraphDef(\test).start(\freq, 440, \amp, 0.125, \gate, 1, \outBus, 1);
GraphDef(\test).stop

//or

GraphDef(\test).start([\freq, 440, \amp, 0.125, \gate, 1, \outBus, 1], s, \addToTail);
GraphDef(\test).stop
```