messaging.nls

brief

This is a simple messaging system for NetLogo agents. It provides the same kind of functionality as nlboris but only supports communication within a single NetLogo world (ie: it does not allow NetLogo agents to communicate with *boris* agents in other subsystems).

Note...
- all incoming messages are queued until an agent chooses to read them (unless their message queues are flushed);
- anyone can send messages but only turtles can receive messages because only turtles have message queues (the code is easy to modify if you want to relax this restriction).

using messaging.nls

import the include file

use the following include statement at the top of your NetLogo code...

```
__includes [ "messaging.nls" ]
```

make-msgQ <name>

`make-msgQ` makes a message queues for a turtle. In the following example we make message queues for a breed called *ants*. Any message receiving entity has a messaging name. This name is supplied as an argument for make-msgQ. In the example the message queue name is the turtles who number prefixed by the breed name...

```
ask ants
[ make-msgQ (word breed who)
]
```

...or alternatively...

```
ants-own [ name ]

ask ants
[ set name (word breed who)
  make-msgQ name
]
```

sending messages

There are two ways to send messages (i) using broadcast (ii) using send-msg. The messages will typically be in the form of a list.

broadcast <breed> <data>
broadcast data to every member of a breed called "*ants*"...

```
broadcast ants some-data
```

send-message <name> <data>
a few examples...

```
;; example 1
send-msg ant1 some-data


;; example 2
let ant [name] of (one-of ants)
send-msg ant some-data


;; example 3 – using the messaging variable #msg-id
let ant [#msg-id] of (one-of ants)
send-msg ant some-data
```

flushing message queues

flush-msgQ <name>

```
;; example 1
flush-msgQ ant1


;; example 2
let ant [name] of (one-of ants)
flush-msgQ ant
```

flush-all-msgQs

```
;; this clears all message queues for all breeds
flush-all-msgQs
```