```
=============================================================================
Concept for MASTER/DEP State Machine ("Load Sensor" for External Prototypes)
=============================================================================

Motivation: (a)
==========  VRML/X3D Browsers may load files asynchronously.
            I.e., if a file refers to other files, then it may happen that
            one file finishes loading before the other files finish.
            Hence it may happen that events passed from a node of one file
            to a node of another file may get lost.
            This is particularly true for events that are sent from the
            initialize() function of a Script node over file borders.
            (b)
            Sometimes, we load parts of the scene dynamically (using the
            Browser.createVrmlFromURL() method). It may happen that the
            loaded part of the scene gets initialized, before we insert it to a
            Group node and before we create dynamic routes to exchange events
            with the loaded part of the scene. Hence the simple solution of just
            outputting an event from the loaded part of the scene, as soon as it
            gets initialized, may fail.

Summary:
=======
     A simple concept is developed, where each external prototype has to contain
     a "dependent" Script node (DEP) and where the loading file (the file which
     contains the proto instances) contains a "master" Script node (MASTER) and
     some routes between the proto instances and the MASTER.
     As soon as all external prototypes are loaded, the MASTER distributes
     a "basicInit" event to all prototypes. Hence the prototypes can exchange
     events arbitrarily during "basic initialization" without loosing events.
     The term "basic initialization" refers to the initialization, which is
     triggered by the mechanisms of the present concept, it is performed AFTER
     the "normal Web3D initialization" (initialize()).
```

```
Scenario I: MASTER and DEP are loaded and initialized synchronously
==========

 MASTER                         DEP
 ______                         ___
   |                             |
(1)|"ping"               "loaded"|(1)
   |----------|       |----------|
   |"loaded"  |       |          |
(2)|<---------|--------    "ping"|
   |          ------------------->|(3)
   |"ping"                 "pong"|
   |----------|       |----------|
   |"pong"    |       |          |
(5)|<---------|--------    "ping"|
   |          ------------------->|(4)
   |                             |
   |                             |
   |                             |
   |                             |
(6)|"basicInit"       "basicInit"|
   |----------------------------->|(7)
   |                             |

     (1)...MASTER and DEP are initialized, send "ping" and "loaded", resp.
     (2)...MASTER receives "loaded" and sends another "ping"
     (3)...DEP receives "ping" and responds with "pong" (first "ping")
     (4)...DEP receives a second "ping" and ignores it
     (5)...MASTER receives "pong" and increments DEP counter
     (6)...as soon as DEP counter reaches the maximal value (now all DEPs are
           loaded and initialized), then MASTER sends "basicInit"
     (7)...all DEPs are "basically initialized" now, they may exchange events
           arbitrarily without any event getting lost
```

Scenario II: MASTER is loaded and initialized first
===========

```
 MASTER
 ______
   |
(1)|"ping"
   |----------|
   |          |
   |          |
   |          ------------------> (2)
   |
   |
   |
   |
   |                            DEP
   |                            ___
   |                             |
   |                     "loaded"|(3)
   |                  |----------|
   |"loaded"          |          |
(4)|<------------------          |
   |"ping"                       |
   |----------|                  |
   |          |                  |
   |          |            "ping"|
   |          ------------------>|(5)
   |                       "pong"|
   |                  |----------|
   |"pong"            |          |
(6)|<------------------          |
   |                             |
   |                             |
   |                             |
   |                             |
(7)|"basicInit"       "basicInit"|
   |---------------------------->|(8)
   |                             |
```

```
     (1)...MASTER is initialized and sends "ping"
     (2)...the first "ping" gets lost
     (3)...DEP is initialized and sends "loaded"
     (4)...MASTER receives "loaded" and sends another "ping"
     (5)...DEP receives "ping" and responds with "pong"
     (6)...MASTER receives "pong" and increments DEP counter
     (7)...as soon as DEP counter reaches the maximal value (now all DEPs are
           loaded and initialized), then MASTER sends "basicInit"
     (8)...all DEPs are "basically initialized" now, they may exchange events
           arbitrarily without any event getting lost
```

Scenario III: DEP is loaded and initialized first

```
                                DEP
                                ___
                                 |
                         "loaded"|(1)
                      |----------|
                      |          |
(2) <------------------          |
                                 |
                                 |
                                 |
                                 |
 MASTER                          |
 ______                          |
   |                             |
(3)|"ping"                       |
   |----------|                  |
   |          |                  |
   |          |            "ping"|
   |          ------------------>|(4)
   |                       "pong"|
   |                  |----------|
   |"pong"            |          |
(5)|<------------------          |
   |                             |
   |                             |
   |                             |
   |                             |
(6)|"basicInit"       "basicInit"|
   |---------------------------->|(7)
   |                             |
```

(1)...DEP is initialized and sends "loaded"
(2)...the "loaded" gets lost
(3)...MASTER is initialized and sends "ping"
(4)...DEP receives "ping" and responds with "pong"
(5)...MASTER receives "pong" and increments DEP counter
(6)...as soon as DEP counter reaches the maximal value (now all DEPs are loaded and initialized), then MASTER sends "basicInit"
(7)...all DEPs are "basically initialized" now, they may exchange events arbitrarily without any event getting lost

Resulting Description of the Concept

(A) The loading file contains a Script node "MASTER" and routes between the MASTER and the proto instances
(B) Each proto declare of the external prototypes contains a Script node "DEP"
(C) In case of nested prototypes, the Scripts in the intermediate prototypes take care about the "MASTER duties" and about the "DEP duties"
(D) Each MASTER has an "initializeOnly" "SFInt32" that indicates the number of dependents ("numDeps")
(E) Each MASTER has an "outputOnly" "SFBool" "sendPing"
(F) Each DEP has an inputOnly" "SFBool" "receivePing"
(G) Each DEP has an "outputOnly" "SFBool" "sendLoaded"
(H) Each MASTER has an "inputOnly" "SFBool" "receiveLoaded"
(I) Each DEP has an "outputOnly" "SFBool" "sendPong"
(J) Each MASTER has an "inputOnly" "SFBool" "receivePong"
(K) Each MASTER has an "outputOnly" "SFBool" "sendBasicInit"
(L) Each DEP has an "inputOnly" "SFBool" "receiveBasicInit"
(M) Each MASTER has an "inputOutput" "SFInt32" "depCounter" "0"
(N) Each DEP has an "inputOutput" "SFBool" "ignorePing" "true"
(O) Behaviour of the MASTER

```
function initialize()
{
  if (numDeps)
    sendPing = true;
  else
    sendBasicInit = true;
}
function receiveLoaded()
{
  sendPing = true;
}
function receivePong()
{
  if (depCounter < numDeps)
  {
    if ((++depCounter) >= numDeps)
    {
      sendBasicInit = true;
    }
  }
}
```

```
(P) Behaviour of the combined MASTER/DEP
      function initialize()
      {
        if (numDeps)
          sendPing = true;
        else
          iAmLoaded();
      }
      function iAmLoaded()
      {
        ignorePing = false;
        sendLoaded = true;
      }
      function receiveLoaded()
      {
        sendPing = true;
      }
      function receivePing()
      {
        if (!ignorePing)
        {
          ignorePing = true;
          sendPong = true;
        }
      }
      function receivePong()
      {
        if (depCounter < numDeps)
        {
          if ((++depCounter) >= numDeps)
          {
            iAmLoaded();
          }
        }
      }
      function receiveBasicInit()
      {
        // TO DO: do my basic initialization here
        sendBasicInit = true;
      }

(Q) Behaviour of the DEP
      function initialize()
      {
        ignorePing = false;
        sendLoaded = true;
      }
      function receivePing()
      {
        if (!ignorePing)
        {
          ignorePing = true;
          sendPong = true;
        }
      }
      function receiveBasicInit()
      {
        // TO DO: do my basic initialization here
      }
```