From valery Thu Mar 24 18:24:18 1994
Subject: CSG: LSD-specification
To: sy@dcs.warwick.ac.uk (Simon Yung)
Date: Thu, 24 Mar 1994 18:24:12 +0000 (GMT)
X-Mailer: ELM [version 2.4 PL20]
Mime-Version: 1.0
Content-Type: text/plain; charset=US-ASCII
Content-Transfer-Encoding: 7bit
Content-Length: 14273
Status: O

INTERACTIVE GEOMETRIC MODELLING SYSTEM

based on

R-FUNCTIONS

LSD-specification.

==========

```
TYPEDEF
              mode  =  {
                          "initialisation"
                          "define_new_gob",
                          "set_space_bound",
                          "set_gob_par",
                          "assign_geom_types",
                          "get_visual_image",
                          "eliminate_gob",
                          "evaluate_po_memb_rel",
                        };

              user_conclusion = {
                           "let's_start",
                           "change_modelling_system"
                           "define_gob",
                           "change_space_bound",
                           "change_gob_par_values",
                           "change_geom_types",
                           "test_point_membership_rel",
                           "change_visual_attr_and_visualize",
                           "eliminate_existing_gob",
                           "let's_finish"
                                 };

               r_model = {
                           "ALPHA = <real_number>",
                           "M = <integer_number>"
                         }

               point_membership_value = {
                                          "outside",
                                          "boundary"
                                          "inside",
                                        };
```

```
            x[i]_geom_type = { "x", "y", "z", "t", "real" };

            rotate_image = { "right", "left", "up", "down" };

================================================================================
AGENT user {
  STATE
       (thought)user_conclusion = "let's_start"
       (bool)modelling_in_progress = FALSE;
  ORACLE
       mode;
  HANDLE
       modelling_in_progress;
       mode;
  DERIVATE
       mode = PROC_user_choose_mode(user_conclusion);
  PROTOCOL
       (user_conclusion == "let"s_start") -->
          modelling_in_progress = TRUE;

       (user's_conclusion == "let's_finish") -->
          modelling_in_progress = FALSE;
--------------------------------------------------------------------

            ; sub-agents of "user" follow further

--------------------------------------------------------------------

   AGENT user_interpretation {
     ORACLE
          mode;
          message;
          point_instance;
          point_membership_value;
          gob_image[gname_current];
          gname_list;
          gname_current;
          gob_descr_fun[*];
          gob_par_list;
          [ xmin[i], xmax[i] ], i=1,...,n;
          x_time;
          alpha,beta;
     HANDLE
          user_conclusion;
     DERIVATE
          LIVE = (modelling_in_progress == TRUE);
          user_conclusion = PROC_user_observe_thinking(
                            message,
                            gname_current,
                            point_instance,
                            point_membership_value,
                            gob_image[gname_current],
                            gob_descr_fun[*],
                            gob_par_list,
                            [ xmin[i],xmax[i], i=1,...,n],
                            x_time,alpha,beta);
        }
-------------------------------------------------------------------------

   AGENT user_init {
    ORACLE
         mode;
         r_model;
         built_in_algebraic_functions;
         built_in_r_functions;
```

```
    HANDLE
         r_model;
         built_in_algebraic_functions;
         built_in_r_functions;
    DERIVATE
         LIVE =  (mode == "initialisation");

          r_model = PROC_user_input(r_model_type)
          built_in_algebraic_functions = PROC_user_input(algebr_fun_type);
          built_in_r_functions = PROC_user_input(r_fun_type);
        }
-------------------------------------------------------------------

   AGENT user_set_space_bound {
     ORACLE
          mode;
          n;
          [ xmin[i], xmax[i] ], i=1,...,n;
     HANDLE
          n;
          [ xmin[i], xmax[i] ], i=1,...,n;
     DERIVATE
          LIVE = (mode == "set_space_bound");

          n = PROC_user_input(n_type);
          [ xmin[i], xmax[i] ] = PROC_user_input(real_type), i=1,...,n;
        }
        }
-------------------------------------------------------------------
   AGENT user_define_gob {
     ORACLE
          mode;
          built_in_algebraic_functions;
          built_in_r_functions;
          gname_list;
          gob_descr_fun[*];
     HANDLE
          gname_current;
          gob_descr_fun[gname_current];
     DERIVATE
          LIVE = (mode == "define_new_gob");

          gname_current = PROC_user_input(gname_type);
          gob_descr_fun[gname_current] = PROC_user_input(descr_fun_type);
        }

-------------------------------------------------------------------
   AGENT user_set_gob_par {
     ORACLE
          mode;
          gob_par_list;
     HANDLE
          new_par_list;
     DERIVATE
          LIVE = (mode == "set_gob_par");

          new_par_list = PROC_user_input(gob_par_type);
        }
-------------------------------------------------------------------
   AGENT user_eval_point_memb_rel {
     ORACLE
          mode;
          gname_list;
     HANDLE
          gname_current;
          x[i], i=1,...,n;
```

```
    DERIVATE
         LIVE = (mode == "evaluate_po_memb_rel" );

         gname_current = PROC_user_input(gname_type);
         [ x[i] ] = PROC_user_input(coord_x[i]_type), i=1,...,n;

        }
-------------------------------------------------------------------------
  AGENT user_assign_geom_types {
    ORACLE
         mode;
         x[i]_geom_type, i=1,...,n;
         x_time;
         x_time_delta, i=1,...,n;
    HANDLE
         x[i]_geom_type, i=1,...,n;
         x_time;
         x_time_delta, i=1,...,n;
    DERIVATE
         LIVE = (mode == "assign_geom_type");

         [ x[i]_geom_type ] = PROC_user_input(x_geom_type), i=1,...,n;
    PROTOCOL
         ( x[i]_geom_type == "t") -->
            {
              x_time = PROC_user_input(real_type)
              x_time_delta = PROC_user_input(delta_type);
            }

        }
-------------------------------------------------------------------------
  AGENT user_get_visual_image {
    ORACLE
         mode;
         gname_list;
         alpha,beta;
         color;
         visual_attr_list;
    HANDLE
         gname_current;
         alpha, beta;
         color;
         visual_attr;
         film;
         rotate_image;
    DERIVATE
         LIVE = (mode == "get_visual_image");

         gname_current = PROC_user_input(gname_type);
         [ alpha, beta ] = PROC_user_input(angles_type);
         color = PROC_user_input(color_type);
         visual_attr_list = PROC_user_input(visual_attr_type);
         film = PROC_user_input(bool_type);
         rotate_image = PROC_user_input(rotate_image_type);

        }

-------------------------------------------------------------------------

  AGENT user_delete_gob {
    ORACLE
         gname_list;
         gob_descr_fun[*];
    HANDLE
         gname_current;
    DERIVATE
```

```
          LIVE = (mode == "eliminate_gob");

          gname_current = PROC_user_input(gname_type);
        }

-------------------------------------------------------------------------------
   }
                      ; the end of "user"

===============================================================================

AGENT modelling {
  STATE
      (string)mode = @;
      (string)r_model = "ALPHA=0."
      (list)built_in_algebraic_functions = { "sqrt","exp","log",
                                             "sin","cos","tang",
                                             "arcsin", "arccos",~abs",
                                             "mod","sign","max","min",...};
      (list)built_in_r_functions = { "|","&","\","~","@"  };
      (list)gname_list = @;
      (string)gname_current = @;
      (bool)gob_exist[*] = FALSE;
      (list)new_par_list = @;
      (string)message = @;
  ORACLE
      modelling_in_progress;
      gob_exist[*];
      point_valid;
      fan_valid[gname];
  HANDLE
      gob_exist[*];
      gob_tree_exist[*];
      message;
  DERIVATE
      (bool)LIVE = modelling_in_progress;
      (string)message = PROC_message(point_valid, fun_valid,...)
  PROTOCOL
      (mode == initialisation") -->
         PROC_init(r_model,built_in_algebraic_functions,
                   built_in_r_functions);

      (mode == "define_new_gob") AND
       (gob_exist[gname_current] == FALSE)  -->
         {
           gob_exist[gname_current] = TRUE;
           gname_list = PROC_append(gname_list,gname_current);
         }

      (mode == "define_new_gob") AND (gob_exist[gname] == TRUE)  -->
        gob_tree_exist[gname] = FALSE;

      (mode == "eliminate_gob") AND
       (gob_exist[gname_current] == TRUE) -->
         {
           gob_exist[gname_current] = FALSE;
           gname_list = PROC_remove(gname_list,gname_current);
         }

      (mode == "set_gob_par") -->
           gob_par_list = PROC_append(gob_par_list,new_par_list);

                                ; sub-agents of "modelling"
                                ; follow further
-------------------------------------------------------------------
```

```
AGENT point_in_space {
  STATE
      (integer)n = 4;
      [ (real)xmin[i] = 0 ], i=1,2,...,n;
      [ (real)xmax[i] = 1 ], i=1,2,...,n;
      [ (real)x[i] = xmin[i] ], i=1,2,...,n;
      (point_type)point_instance = @;
      (bool)point_valid = @;
      x[1]_geom_type = "x";
      x[2]_geom_type = "y";
      x[3]_geom_type = "z";
      x[4]_geom_type = "t"
      x[i]_geom_type = "real", i=5,...,n;
      x_time = xmin[4];
      x_time_delta = (xmax[4]-xmin[4])/10.;
  DERIVATE
      point_instance = [ x[1], x[2],... x[n] ];

      point_valid = AND ( xmin[i] <= x[i] <= xmax[i] ); i=1,2,...,n

  }
```

```
AGENT geometric_object [gname] {
  STATE
     (symbolic_rep)gob_descr_fun[gname] = @;
     (real)gob_par_list[gname] = @;
     (csg_tree_rep)gob_csg_tree[gname] = @;
     (screen_picture)gob_image[gname] = @;
     (real)gob_value[gname] = @;
     (bool)gob_tree_exist[gname] = FALSE;
  ORACLE
     mode;
     point_instance;
  DERIVATE
     (bool)LIVE = gob_exist[gname];
     gob_implicit_value[gname] =
       PROC_gob_descr_fun_eval(point_instance,gob_tree[gname],
                               gob_par_list);
  }
```

```
AGENT r_conversion {
  STATE
    (bool)fun_valid[gname_current] = FALSE;
  ORACLE
    gob_descr_fun[gname_current];
    built_in_algebraic_functions;
    built_in_r_functions;
  HANDLE
    gob_tree_exist[gname_current];
    gob_csg_tree[gname_current];
    fun_valid[gname_current];
  DERIVATE
    (bool)LIVE = (mode == "define_new_gob")
    fun_valid[gname_current] = PROC_test_fun(gob_descr_fun[gname_current]);
  PROTOCOL

    (fun_valid[gname_current] == TRUE) AND
      (gob_tree_exist[gname_current] == FALSE) -->
          {
            gob_csg_tree[gname_current] =
              PROC_create_gob_tree(gob_descr_fun[gname_current],
                                   built_in_algebraic_functions,
```

```
                                        built_in_r_functions);
              gob_tree_exist[gname] = TRUE;
            }
    }
```

```
  AGENT point_membership_relation {
    STATE
      (string)point_membership_value = @;
    ORACLE
      gob_implicit_value[gname_current];
    DERIVATE
      (bool)LIVE = ( mode == "evaluate_po_memb_rel" ) OR
                   mode == "get_visual_image" );

      point_membership_value =
       PROC_predicate_eval(gob_implicit_value[gname_current]);

    }
```

```
  AGENT visualisation {
    STATE
      (list)visual_attr_list = ...;
      (degree)alpha = -30.;
      (degree)beta = 30.;
      (delta)alpha = 10.;
      (delta)beta = 10;
      (color_type)color = monochrome;
      (bool)film = FALSE;
      (direction)rotate_image = "left";
    ORACLE
      gob_csg_tree[gname_current];
      point_membership_value;
      x[i]_type, i=1,...,n;
      xmin[i], i=1,...,n;
      xmax[i], i=1,...,n;
      x_time;
      x_time_delta;
    HANDLE
      gob_image[gname_current];
      x[i], i=1,...,n;
      x_time_delta;
    DERIVATE
      (bool)LIVE = ( mode=="get_visual_image" );
      gob_image[gname_current] =  PROC_visualize(
                                        gob_csg_tree[gname_current],
                                        [ x[i]_geom_type,i=1,...,n ],
                                        x_time,
                                        color,alpha,beta,visual_attr_list,);
    PROTOCOL
      ( film == TRUE ) -->
          PROC_save_frame( gob_image[gname_current] );

      ( x[i]_geom_type == "t" ) AND
       ( xmin[i] <= x_time <= xmax[i] ) -->
         x_time = x_time + xtime_delta;

      ( rotate_image == "left" )  -->
        alpha = alpha + delta_alpha;

      ( rotate_image == "right" ) -->
        alpha = alpha - delta_alpha;

      ( rotate_image == "up" ) -->
```

```
        beta = beta - delta_beta;

      ( rotate_image == "down" )  -->
        beta = beta + delta_beta;
    }
 }
                              ; the end of "modelling"
=============================================================================
```