```
classdef remApi

    properties
        libName;
        hFile;
    end
    properties (Constant)

        % Scene object types
        sim_object_shape_type           =0;
        sim_object_joint_type           =1;
        sim_object_graph_type           =2;
        sim_object_camera_type          =3;
        sim_object_dummy_type           =4;
        sim_object_proximitysensor_type =5;
        sim_object_reserved1            =6;
        sim_object_reserved2            =7;
        sim_object_path_type            =8;
        sim_object_visionsensor_type    =9;
        sim_object_volume_type          =10;
        sim_object_mill_type            =11;
        sim_object_forcesensor_type     =12;
        sim_object_light_type           =13;
        sim_object_mirror_type          =14;

        %General object types
        sim_appobj_object_type          =109;
        sim_appobj_collision_type       =110;
        sim_appobj_distance_type        =111;
        sim_appobj_simulation_type      =112;
        sim_appobj_ik_type              =113;
        sim_appobj_constraintsolver_type=114;
        sim_appobj_collection_type      =115;
        sim_appobj_ui_type              =116;
        sim_appobj_script_type          =117;
        sim_appobj_pathplanning_type    =118;
        sim_appobj_RESERVED_type        =119;
        sim_appobj_texture_type         =120;

        % Ik calculation methods
        sim_ik_pseudo_inverse_method        =0;
        sim_ik_damped_least_squares_method  =1;
        sim_ik_jacobian_transpose_method    =2;
```

```
% Ik constraints
sim_ik_x_constraint         =1;
sim_ik_y_constraint         =2;
sim_ik_z_constraint         =4;
sim_ik_alpha_beta_constraint=8;
sim_ik_gamma_constraint     =16;
sim_ik_avoidance_constraint =64;

% Ik calculation results
sim_ikresult_not_performed  =0;
sim_ikresult_success        =1;
sim_ikresult_fail           =2;

% Scene object sub-types
sim_light_omnidirectional_subtype   =1;
sim_light_spot_subtype              =2;
sim_light_directional_subtype       =3;
sim_joint_revolute_subtype          =10;
sim_joint_prismatic_subtype         =11;
sim_joint_spherical_subtype         =12;
sim_shape_simpleshape_subtype       =20;
sim_shape_multishape_subtype        =21;
sim_proximitysensor_pyramid_subtype =30;
sim_proximitysensor_cylinder_subtype=31;
sim_proximitysensor_disc_subtype    =32;
sim_proximitysensor_cone_subtype    =33;
sim_proximitysensor_ray_subtype     =34;
sim_mill_pyramid_subtype            =40;
sim_mill_cylinder_subtype           =41;
sim_mill_disc_subtype               =42;
sim_mill_cone_subtype               =42;
sim_object_no_subtype               =200;


%Scene object main properties
sim_objectspecialproperty_collidable                =1;
sim_objectspecialproperty_measurable                =2;
sim_objectspecialproperty_detectable_ultrasonic     =16;
sim_objectspecialproperty_detectable_infrared       =32;
sim_objectspecialproperty_detectable_laser          =64;
sim_objectspecialproperty_detectable_inductive      =128;
sim_objectspecialproperty_detectable_capacitive     =256;
```

```
sim_objectspecialproperty_renderable                =512;
sim_objectspecialproperty_detectable_all            =496;
sim_objectspecialproperty_cuttable                  =1024;
sim_objectspecialproperty_pathplanning_ignored      =2048;

% Model properties
sim_modelproperty_not_collidable            =1;
sim_modelproperty_not_measurable            =2;
sim_modelproperty_not_renderable            =4;
sim_modelproperty_not_detectable            =8;
sim_modelproperty_not_cuttable              =16;
sim_modelproperty_not_dynamic               =32;
sim_modelproperty_not_respondable           =64;
sim_modelproperty_not_reset                 =128;
sim_modelproperty_not_visible               =256;
sim_modelproperty_not_model                 =61440;


% Check the documentation instead of comments below!!
sim_message_ui_button_state_change  =0;
sim_message_reserved9               =1;
sim_message_object_selection_changed=2;
sim_message_reserved10              =3;
sim_message_model_loaded            =4;
sim_message_reserved11              =5;
sim_message_keypress                =6;
sim_message_bannerclicked           =7;
sim_message_for_c_api_only_start    =256;
sim_message_reserved1               =257;
sim_message_reserved2               =258;
sim_message_reserved3               =259;
sim_message_eventcallback_scenesave     =260;
sim_message_eventcallback_modelsave     =261;
sim_message_eventcallback_moduleopen    =262;
sim_message_eventcallback_modulehandle  =263;
sim_message_eventcallback_moduleclose   =264;
sim_message_reserved4               =265;
sim_message_reserved5               =266;
sim_message_reserved6               =267;
sim_message_reserved7               =268;
sim_message_eventcallback_instancepass  =269;
sim_message_eventcallback_broadcast     =270;
sim_message_eventcallback_imagefilter_enumreset =271;
```

```
sim_message_eventcallback_imagefilter_enumerate      =272;
sim_message_eventcallback_imagefilter_adjustparams   =273;
sim_message_eventcallback_imagefilter_reserved       =274;
sim_message_eventcallback_imagefilter_process        =275;
sim_message_eventcallback_reserved1                  =276;
sim_message_eventcallback_reserved2                  =277;
sim_message_eventcallback_reserved3                  =278;
sim_message_eventcallback_reserved4                  =279;
sim_message_eventcallback_abouttoundo                =280;
sim_message_eventcallback_undoperformed              =281;
sim_message_eventcallback_abouttoredo                =282;
sim_message_eventcallback_redoperformed              =283;
sim_message_eventcallback_scripticondblclick         =284;
sim_message_eventcallback_simulationabouttostart     =285;
sim_message_eventcallback_simulationended            =286;
sim_message_eventcallback_reserved5                  =287;
sim_message_eventcallback_keypress                   =288;
sim_message_eventcallback_modulehandleinsensingpart  =289;
sim_message_eventcallback_renderingpass              =290;
sim_message_eventcallback_bannerclicked              =291;
sim_message_eventcallback_menuitemselected           =292;
sim_message_eventcallback_refreshdialogs             =293;
sim_message_eventcallback_sceneloaded                =294;
sim_message_eventcallback_modelloaded                =295;
sim_message_eventcallback_instanceswitch             =296;
sim_message_eventcallback_guipass                    =297;
sim_message_eventcallback_mainscriptabouttobecalled  =298;
sim_message_eventcallback_rmlposition                =299;
sim_message_eventcallback_rmlvelocity                =300;

sim_message_simulation_start_resume_request          =4096;
sim_message_simulation_pause_request                 =4097;
sim_message_simulation_stop_request                  =4098;

% Scene object properties
sim_objectproperty_collapsed               =16;
sim_objectproperty_selectable              =32;
sim_objectproperty_reserved7               =64;
sim_objectproperty_selectmodelbaseinstead  =128;
sim_objectproperty_dontshowasinsidemodel   =256;
sim_objectproperty_canupdatedna            =1024;
sim_objectproperty_selectinvisible         =2048;
sim_objectproperty_depthinvisible          =4096;
```

```
% type of arguments (input and output) for custom lua commands
sim_lua_arg_nil     =0;
sim_lua_arg_bool    =1;
sim_lua_arg_int     =2;
sim_lua_arg_float   =3;
sim_lua_arg_string  =4;
sim_lua_arg_invalid =5;
sim_lua_arg_table   =8;

% custom user interface properties
sim_ui_property_visible                    =1;
sim_ui_property_visibleduringsimulationonly =2;
sim_ui_property_moveable                   =4;
sim_ui_property_relativetoleftborder       =8;
sim_ui_property_relativetotopborder        =16;
sim_ui_property_fixedwidthfont             =32;
sim_ui_property_systemblock                =64;
sim_ui_property_settocenter                =128;
sim_ui_property_rolledup                   =256;
sim_ui_property_selectassociatedobject     =512;
sim_ui_property_visiblewhenobjectselected  =1024;


% button properties
sim_buttonproperty_button              =0;
sim_buttonproperty_label               =1;
sim_buttonproperty_slider              =2;
sim_buttonproperty_editbox             =3;
sim_buttonproperty_staydown            =8;
sim_buttonproperty_enabled             =16;
sim_buttonproperty_borderless          =32;
sim_buttonproperty_horizontallycentered =64;
sim_buttonproperty_ignoremouse         =128;
sim_buttonproperty_isdown              =256;
sim_buttonproperty_transparent         =512;
sim_buttonproperty_nobackgroundcolor   =1024;
sim_buttonproperty_rollupaction        =2048;
sim_buttonproperty_closeaction         =4096;
sim_buttonproperty_verticallycentered  =8192;
sim_buttonproperty_downupevent         =16384;
```

```
% Simulation status
sim_simulation_stopped                  =0;
sim_simulation_paused                   =8;
sim_simulation_advancing                =16;
sim_simulation_advancing_firstafterstop =16;
sim_simulation_advancing_running        =17;
sim_simulation_advancing_lastbeforepause =19;
sim_simulation_advancing_firstafterpause =20;
sim_simulation_advancing_abouttostop    =21;
sim_simulation_advancing_lastbeforestop =22;


% Script execution result (first return value)
sim_script_no_error             =0;
sim_script_main_script_nonexistent =1;
sim_script_main_script_not_called =2;
sim_script_reentrance_error     =4;
sim_script_lua_error            =8;
sim_script_call_error           =16;


% Script types
sim_scripttype_mainscript   =0;
sim_scripttype_childscript  =1;
sim_scripttype_jointctrlcallback =4;
sim_scripttype_contactcallback =5;
sim_scripttype_customizationscript =6;
sim_scripttype_generalcallback =7;

% API call error messages
sim_api_errormessage_ignore =0;
sim_api_errormessage_report =1;
sim_api_errormessage_output =2;


% special argument of some functions
sim_handle_all                  =-2;
sim_handle_all_except_explicit  =-3;
sim_handle_self                 =-4;
sim_handle_main_script          =-5;
sim_handle_tree                 =-6;
```

```
sim_handle_chain                =-7;
sim_handle_single               =-8;
sim_handle_default              =-9;
sim_handle_all_except_self      =-10;
sim_handle_parent               =-11;

% special handle flags
sim_handleflag_assembly         =4194304;
sim_handleflag_model            =8388608;


% distance calculation methods
sim_distcalcmethod_dl           =0;
sim_distcalcmethod_dac          =1;
sim_distcalcmethod_max_dl_dac   =2;
sim_distcalcmethod_dl_and_dac   =3;
sim_distcalcmethod_sqrt_dl2_and_dac2=4;
sim_distcalcmethod_dl_if_nonzero    =5;
sim_distcalcmethod_dac_if_nonzero   =6;


% Generic dialog styles
sim_dlgstyle_message        =0;
sim_dlgstyle_input          =1;
sim_dlgstyle_ok             =2;
sim_dlgstyle_ok_cancel      =3;
sim_dlgstyle_yes_no         =4;
sim_dlgstyle_dont_center    =32;

% Generic dialog return values
sim_dlgret_still_open   =0;
sim_dlgret_ok           =1;
sim_dlgret_cancel       =2;
sim_dlgret_yes          =3;
sim_dlgret_no           =4;


% Path properties
sim_pathproperty_show_line                  =1;
sim_pathproperty_show_orientation           =2;
sim_pathproperty_closed_path                =4;
sim_pathproperty_automatic_orientation      =8;
sim_pathproperty_invert_velocity            =16;
```

```
sim_pathproperty_infinite_acceleration            =32;
sim_pathproperty_flat_path                        =64;
sim_pathproperty_show_position                    =128;
sim_pathproperty_auto_velocity_profile_translation  =256;
sim_pathproperty_auto_velocity_profile_rotation     =512;
sim_pathproperty_endpoints_at_zero                =1024;
sim_pathproperty_keep_x_up                        =2048;


% drawing objects
sim_drawing_points         =0;
sim_drawing_lines          =1;
sim_drawing_triangles      =2;
sim_drawing_trianglepoints  =3;
sim_drawing_quadpoints     =4;
sim_drawing_discpoints     =5;
sim_drawing_cubepoints     =6;
sim_drawing_spherepoints   =7;

sim_drawing_itemcolors             =32;
sim_drawing_vertexcolors           =64;
sim_drawing_itemsizes              =128;
sim_drawing_backfaceculling        =256;
sim_drawing_wireframe              =512;
sim_drawing_painttag               =1024;
sim_drawing_followparentvisibility  =2048;
sim_drawing_cyclic                 =4096;
sim_drawing_50percenttransparency   =8192;
sim_drawing_25percenttransparency   =16384;
sim_drawing_12percenttransparency   =32768;
sim_drawing_emissioncolor          =65536;
sim_drawing_facingcamera           =131072;
sim_drawing_overlay                =262144;
sim_drawing_itemtransparency       =524288;

% banner values
sim_banner_left                  =1;
sim_banner_right                 =2;
sim_banner_nobackground          =4;
sim_banner_overlay               =8;
sim_banner_followparentvisibility   =16;
sim_banner_clickselectsparent       =32;
sim_banner_clicktriggersevent       =64;
```

```
sim_banner_facingcamera            =128;
sim_banner_fullyfacingcamera       =256;
sim_banner_backfaceculling         =512;
sim_banner_keepsamesize            =1024;
sim_banner_bitmapfont              =2048;

% particle objects
sim_particle_points1       =0;
sim_particle_points2       =1;
sim_particle_points4       =2;
sim_particle_roughspheres  =3;
sim_particle_spheres       =4;

sim_particle_respondable1to4       =32;
sim_particle_respondable5to8       =64;
sim_particle_particlerespondable   =128;
sim_particle_ignoresgravity        =256;
sim_particle_invisible             =512;
sim_particle_itemsizes             =1024;
sim_particle_itemdensities         =2048;
sim_particle_itemcolors            =4096;
sim_particle_cyclic                =8192;
sim_particle_emissioncolor         =16384;
sim_particle_water                 =32768;
sim_particle_painttag              =65536;

% custom user interface menu attributes
sim_ui_menu_title      =1;
sim_ui_menu_minimize   =2;
sim_ui_menu_close      =4;
sim_ui_menu_systemblock =8;



% Boolean parameters
sim_boolparam_hierarchy_visible               =0;
sim_boolparam_console_visible                 =1;
sim_boolparam_collision_handling_enabled      =2;
sim_boolparam_distance_handling_enabled       =3;
sim_boolparam_ik_handling_enabled             =4;
sim_boolparam_gcs_handling_enabled            =5;
sim_boolparam_dynamics_handling_enabled       =6;
sim_boolparam_joint_motion_handling_enabled   =7;
```

```
sim_boolparam_path_motion_handling_enabled      =8;
sim_boolparam_proximity_sensor_handling_enabled =9;
sim_boolparam_vision_sensor_handling_enabled    =10;
sim_boolparam_mill_handling_enabled             =11;
sim_boolparam_browser_visible                   =12;
sim_boolparam_scene_and_model_load_messages     =13;
sim_reserved0                                   =14;
sim_boolparam_shape_textures_are_visible        =15;
sim_boolparam_display_enabled                   =16;
sim_boolparam_infotext_visible                  =17;
sim_boolparam_statustext_open                   =18;
sim_boolparam_fog_enabled                       =19;
sim_boolparam_rml2_available                    =20;
sim_boolparam_rml4_available                    =21;
sim_boolparam_mirrors_enabled                   =22;
sim_boolparam_aux_clip_planes_enabled           =23;
sim_boolparam_full_model_copy_from_api          =24;
sim_boolparam_realtime_simulation               =25;
sim_boolparam_force_show_wireless_emission      =27;
sim_boolparam_force_show_wireless_reception     =28;
sim_boolparam_video_recording_triggered         =29;
sim_boolparam_threaded_rendering_enabled        =32;
sim_boolparam_fullscreen                        =33;
sim_boolparam_headless                          =34;
sim_boolparam_hierarchy_toolbarbutton_enabled   =35;
sim_boolparam_browser_toolbarbutton_enabled     =36;
sim_boolparam_objectshift_toolbarbutton_enabled =37;
sim_boolparam_objectrotate_toolbarbutton_enabled=38;
sim_boolparam_force_calcstruct_all_visible      =39;
sim_boolparam_force_calcstruct_all              =40;
sim_boolparam_exit_request                      =41;
sim_boolparam_play_toolbarbutton_enabled        =42;
sim_boolparam_pause_toolbarbutton_enabled       =43;
sim_boolparam_stop_toolbarbutton_enabled        =44;
sim_boolparam_waiting_for_trigger       =45;


% Integer parameters
sim_intparam_error_report_mode      =0;
sim_intparam_program_version        =1;
sim_intparam_instance_count         =2;
sim_intparam_custom_cmd_start_id    =3;
sim_intparam_compilation_version    =4;
```

```
sim_intparam_current_page          =5;
sim_intparam_flymode_camera_handle  =6;
sim_intparam_dynamic_step_divider   =7;
sim_intparam_dynamic_engine         =8;
sim_intparam_server_port_start      =9;
sim_intparam_server_port_range      =10;
sim_intparam_visible_layers         =11;
sim_intparam_infotext_style         =12;
sim_intparam_settings               =13;
sim_intparam_edit_mode_type         =14;
sim_intparam_server_port_next       =15;
sim_intparam_qt_version             =16;
sim_intparam_event_flags_read       =17;
sim_intparam_event_flags_read_clear =18;
sim_intparam_platform               =19;
sim_intparam_scene_unique_id        =20;
sim_intparam_work_thread_count      =21;
sim_intparam_mouse_x                =22;
sim_intparam_mouse_y                =23;
sim_intparam_core_count             =24;
sim_intparam_work_thread_calc_time_ms =25;
sim_intparam_idle_fps               =26;
sim_intparam_prox_sensor_select_down =27;
sim_intparam_prox_sensor_select_up  =28;
sim_intparam_stop_request_counter   =29;
sim_intparam_program_revision       =30;
sim_intparam_mouse_buttons          =31;
sim_intparam_dynamic_warning_disabled_mask =32;
sim_intparam_simulation_warning_disabled_mask =33;
sim_intparam_scene_index            =34;
sim_intparam_motionplanning_seed    =35;
sim_intparam_speedmodifier          =36;

% Float parameters
sim_floatparam_rand                 =0;
sim_floatparam_simulation_time_step =1;
sim_floatparam_stereo_distance      =2;

% String parameters
sim_stringparam_application_path    =0;
sim_stringparam_video_filename      =1;
sim_stringparam_app_arg1            =2;
sim_stringparam_app_arg2            =3;
```

```
sim_stringparam_app_arg3          =4;
sim_stringparam_app_arg4          =5;
sim_stringparam_app_arg5          =6;
sim_stringparam_app_arg6          =7;
sim_stringparam_app_arg7          =8;
sim_stringparam_app_arg8          =9;
sim_stringparam_app_arg9          =10;
sim_stringparam_scene_path_and_name =13;

% Array parameters
sim_arrayparam_gravity         =0;
sim_arrayparam_fog             =1;
sim_arrayparam_fog_color       =2;
sim_arrayparam_background_color1=3;
sim_arrayparam_background_color2=4;
sim_arrayparam_ambient_light    =5;
sim_arrayparam_random_euler     =6;

sim_objintparam_visibility_layer= 10;
sim_objfloatparam_abs_x_velocity= 11;
sim_objfloatparam_abs_y_velocity= 12;
sim_objfloatparam_abs_z_velocity= 13;
sim_objfloatparam_abs_rot_velocity= 14;
sim_objfloatparam_objbbox_min_x= 15;
sim_objfloatparam_objbbox_min_y= 16;
sim_objfloatparam_objbbox_min_z= 17;
sim_objfloatparam_objbbox_max_x= 18;
sim_objfloatparam_objbbox_max_y= 19;
sim_objfloatparam_objbbox_max_z= 20;
sim_objfloatparam_modelbbox_min_x= 21;
sim_objfloatparam_modelbbox_min_y= 22;
sim_objfloatparam_modelbbox_min_z= 23;
sim_objfloatparam_modelbbox_max_x= 24;
sim_objfloatparam_modelbbox_max_y= 25;
sim_objfloatparam_modelbbox_max_z= 26;
sim_objintparam_collection_self_collision_indicator= 27;
sim_objfloatparam_transparency_offset= 28;
sim_objintparam_child_role= 29;
sim_objintparam_parent_role= 30;
sim_objintparam_manipulation_permissions= 31;
sim_objintparam_illumination_handle= 32;

sim_visionfloatparam_near_clipping= 1000;
```

sim_visionfloatparam_far_clipping= 1001;
sim_visionintparam_resolution_x= 1002;
sim_visionintparam_resolution_y= 1003;
sim_visionfloatparam_perspective_angle= 1004;
sim_visionfloatparam_ortho_size= 1005;
sim_visionintparam_disabled_light_components= 1006;
sim_visionintparam_rendering_attributes= 1007;
sim_visionintparam_entity_to_render= 1008;
sim_visionintparam_windowed_size_x= 1009;
sim_visionintparam_windowed_size_y= 1010;
sim_visionintparam_windowed_pos_x= 1011;
sim_visionintparam_windowed_pos_y= 1012;
sim_visionintparam_pov_focal_blur= 1013;
sim_visionfloatparam_pov_blur_distance= 1014;
sim_visionfloatparam_pov_aperture= 1015;
sim_visionintparam_pov_blur_sampled= 1016;
sim_visionintparam_render_mode= 1017;

sim_jointintparam_motor_enabled= 2000;
sim_jointintparam_ctrl_enabled= 2001;
sim_jointfloatparam_pid_p= 2002;
sim_jointfloatparam_pid_i= 2003;
sim_jointfloatparam_pid_d= 2004;
sim_jointfloatparam_intrinsic_x= 2005;
sim_jointfloatparam_intrinsic_y= 2006;
sim_jointfloatparam_intrinsic_z= 2007;
sim_jointfloatparam_intrinsic_qx= 2008;
sim_jointfloatparam_intrinsic_qy= 2009;
sim_jointfloatparam_intrinsic_qz= 2010;
sim_jointfloatparam_intrinsic_qw= 2011;
sim_jointfloatparam_velocity= 2012;
sim_jointfloatparam_spherical_qx= 2013;
sim_jointfloatparam_spherical_qy= 2014;
sim_jointfloatparam_spherical_qz= 2015;
sim_jointfloatparam_spherical_qw= 2016;
sim_jointfloatparam_upper_limit= 2017;
sim_jointfloatparam_kc_k= 2018;
sim_jointfloatparam_kc_c= 2019;
sim_jointfloatparam_ik_weight= 2021;
sim_jointfloatparam_error_x= 2022;
sim_jointfloatparam_error_y= 2023;
sim_jointfloatparam_error_z= 2024;
sim_jointfloatparam_error_a= 2025;

```
sim_jointfloatparam_error_b= 2026;
sim_jointfloatparam_error_g= 2027;
sim_jointfloatparam_error_pos= 2028;
sim_jointfloatparam_error_angle= 2029;
sim_jointintparam_velocity_lock= 2030;
sim_jointintparam_vortex_dep_handle= 2031;
sim_jointfloatparam_vortex_dep_multiplication= 2032;
sim_jointfloatparam_vortex_dep_offset= 2033;

sim_shapefloatparam_init_velocity_x= 3000;
sim_shapefloatparam_init_velocity_y= 3001;
sim_shapefloatparam_init_velocity_z= 3002;
sim_shapeintparam_static= 3003;
sim_shapeintparam_respondable= 3004;
sim_shapefloatparam_mass= 3005;
sim_shapefloatparam_texture_x= 3006;
sim_shapefloatparam_texture_y= 3007;
sim_shapefloatparam_texture_z= 3008;
sim_shapefloatparam_texture_a= 3009;
sim_shapefloatparam_texture_b= 3010;
sim_shapefloatparam_texture_g= 3011;
sim_shapefloatparam_texture_scaling_x= 3012;
sim_shapefloatparam_texture_scaling_y= 3013;
sim_shapeintparam_culling= 3014;
sim_shapeintparam_wireframe= 3015;
sim_shapeintparam_compound= 3016;
sim_shapeintparam_convex= 3017;
sim_shapeintparam_convex_check= 3018;
sim_shapeintparam_respondable_mask= 3019;
sim_shapefloatparam_init_velocity_a= 3020;
sim_shapefloatparam_init_velocity_b= 3021;
sim_shapefloatparam_init_velocity_g= 3022;
sim_shapestringparam_color_name= 3023;
sim_shapeintparam_edge_visibility= 3024;
sim_shapefloatparam_shading_angle= 3025;
sim_shapefloatparam_edge_angle= 3026;
sim_shapeintparam_edge_borders_hidden= 3027;

sim_proxintparam_ray_invisibility= 4000;

sim_forcefloatparam_error_x= 5000;
sim_forcefloatparam_error_y= 5001;
sim_forcefloatparam_error_z= 5002;
```

```
sim_forcefloatparam_error_a= 5003;
sim_forcefloatparam_error_b= 5004;
sim_forcefloatparam_error_g= 5005;
sim_forcefloatparam_error_pos= 5006;
sim_forcefloatparam_error_angle= 5007;

sim_lightintparam_pov_casts_shadows= 8000;

sim_cameraintparam_disabled_light_components= 9000;
sim_camerafloatparam_perspective_angle= 9001;
sim_camerafloatparam_ortho_size= 9002;
sim_cameraintparam_rendering_attributes= 9003;
sim_cameraintparam_pov_focal_blur= 9004;
sim_camerafloatparam_pov_blur_distance= 9005;
sim_camerafloatparam_pov_aperture= 9006;
sim_cameraintparam_pov_blur_samples= 9007;

sim_dummyintparam_link_type= 10000;

sim_mirrorfloatparam_width= 12000;
sim_mirrorfloatparam_height= 12001;
sim_mirrorfloatparam_reflectance= 12002;
sim_mirrorintparam_enable= 12003;

sim_pplanfloatparam_x_min= 20000;
sim_pplanfloatparam_x_range= 20001;
sim_pplanfloatparam_y_min= 20002;
sim_pplanfloatparam_y_range= 20003;
sim_pplanfloatparam_z_min= 20004;
sim_pplanfloatparam_z_range= 20005;
sim_pplanfloatparam_delta_min= 20006;
sim_pplanfloatparam_delta_range= 20007;

sim_mplanintparam_nodes_computed= 25000;
sim_mplanintparam_prepare_nodes= 25001;
sim_mplanintparam_clear_nodes= 25002;

% User interface elements
sim_gui_menubar                 =1;
sim_gui_popups                  =2;
sim_gui_toolbar1                =4;
sim_gui_toolbar2                =8;
sim_gui_hierarchy               =16;
```

```
sim_gui_infobar                 =32;
sim_gui_statusbar               =64;
sim_gui_scripteditor            =128;
sim_gui_scriptsimulationparameters  =256;
sim_gui_dialogs                 =512;
sim_gui_browser                 =1024;
sim_gui_all                     =65535;


% Joint modes
sim_jointmode_passive       =0;
sim_jointmode_motion        =1;
sim_jointmode_ik            =2;
sim_jointmode_ikdependent   =3;
sim_jointmode_dependent     =4;
sim_jointmode_force         =5;


% Navigation and selection modes with the mouse.
sim_navigation_passive              =0;
sim_navigation_camerashift          =1;
sim_navigation_camerarotate         =2;
sim_navigation_camerazoom           =3;
sim_navigation_cameratilt           =4;
sim_navigation_cameraangle          =5;
sim_navigation_camerafly            =6;
sim_navigation_objectshift          =7;
sim_navigation_objectrotate         =8;
sim_navigation_reserved2            =9;
sim_navigation_reserved3            =10;
sim_navigation_jointpathtest        =11;
sim_navigation_ikmanip              =12;
sim_navigation_objectmultipleselection  =13;

sim_navigation_reserved4            =256;
sim_navigation_clickselection       =512;
sim_navigation_ctrlselection        =1024;
sim_navigation_shiftselection       =2048;
sim_navigation_camerazoomwheel      =4096;
sim_navigation_camerarotaterightbutton  =8192;


% Remote API message header structure
```

```
simx_headeroffset_crc          =0;
simx_headeroffset_version      =2;
simx_headeroffset_message_id   =3;
simx_headeroffset_client_time  =7;
simx_headeroffset_server_time  =11;
simx_headeroffset_scene_id     =15;
simx_headeroffset_server_state =17;

% Remote API command header
simx_cmdheaderoffset_mem_size      =0;
simx_cmdheaderoffset_full_mem_size =4;
simx_cmdheaderoffset_pdata_offset0 =8;
simx_cmdheaderoffset_pdata_offset1 =10;
simx_cmdheaderoffset_cmd           =14;
simx_cmdheaderoffset_delay_or_split =18;
simx_cmdheaderoffset_sim_time      =20;
simx_cmdheaderoffset_status        =24;
simx_cmdheaderoffset_reserved      =25;


% Regular operation modes
simx_opmode_oneshot           =0;
simx_opmode_blocking          =65536;
simx_opmode_oneshot_wait      =65536;
simx_opmode_continuous        =131072;
simx_opmode_streaming         =131072;

% Operation modes for heavy data
simx_opmode_oneshot_split     =196608;
simx_opmode_continuous_split  =262144;
simx_opmode_streaming_split   =262144;

% Special operation modes
simx_opmode_discontinue       =327680;
simx_opmode_buffer            =393216;
simx_opmode_remove            =458752;


% Command return codes
simx_return_ok                  =0;
simx_return_novalue_flag        =1;
simx_return_timeout_flag        =2;
simx_return_illegal_opmode_flag =4;
```

```
    simx_return_remote_error_flag       =8;
    simx_return_split_progress_flag     =16;
    simx_return_local_error_flag        =32;
    simx_return_initialize_error_flag   =64;

    % Following for backward compatibility (same as above)
    simx_error_noerror                  =0;
    simx_error_novalue_flag             =1;
    simx_error_timeout_flag             =2;
    simx_error_illegal_opmode_flag      =4;
    simx_error_remote_error_flag        =8;
    simx_error_split_progress_flag      =16;
    simx_error_local_error_flag         =32;
    simx_error_initialize_error_flag    =64;

end
methods

    %constructor
    function obj = remApi(libname,hfile)
        obj.libName = libname;
        %fprintf('Running Matlab %s\n',computer('arch'));
        disp('Note: always make sure you use the corresponding remoteApi library');
        disp('(i.e. 32bit Matlab will not work with 64bit remoteApi, and vice-versa)');
        if ~libisloaded(obj.libName)
            if exist('hfile','var')
                obj.hFile = hfile;
                loadlibrary(obj.libName,obj.hFile);
            else
                loadlibrary(obj.libName,@remoteApiProto);
            end
        end
    end

    %destructor

    function delete(obj)
        % we keep the library in memory for now          unloadlibrary(obj.libName);
    end


    %api methods
```

```
        function [rtn] =
simxStart(obj,server,port,waitUntilConnected,doNotReconnectOnceDisconnected,timeOutInMs,
commThreadCycleInMs)
            server_intval = int8([server,0]);
            server_ptr = libpointer('int8Ptr',server_intval);
            rtn =
calllib(obj.libName,'simxStart',server_ptr,port,uint8(waitUntilConnected),uint8(doNotReconnectO
nceDisconnected),timeOutInMs,commThreadCycleInMs);
        end

        function simxFinish(obj,clientID)
            calllib(obj.libName,'simxFinish',clientID);
        end

        function rtn = simxAddStatusbarMessage(obj,clientID,message,operationmode)
            message_ = libpointer('int8Ptr',[int8(message) 0]);
            operationmode_ = int32(operationmode);

            [rtn message_] =
calllib(obj.libName,'simxAddStatusbarMessage',clientID,message_,operationmode_);
        end

        function rtn = simxAuxiliaryConsoleClose(obj,clientID,console, operationmode)
            console_ = int32(console);
            operationmode_ = int32(operationmode);

            rtn = calllib(obj.libName,'simxAuxiliaryConsoleClose',clientID,console_,operationmode_)
        end

        function rtn = simxAuxiliaryConsolePrint(obj,clientID,console,text,operationmode)
            console_ = int32(console);
            operationmode_ = int32(operationmode);

            if text
                text_ = libpointer('int8Ptr',[int8(text) 0]);
            else
                text_ = [];
            end

            rtn =
calllib(obj.libName,'simxAuxiliaryConsolePrint',clientID,console_,text_,operationmode_);
        end
```

```
        function rtn =
simxAuxiliaryConsoleShow(obj,clientID,consoleHandle,showState,operationMode)
            consoleHandle_ = int32(consoleHandle);
            showState_ = uint8(showState);
            operationMode_ = int32(operationMode);

            rtn =
calllib(obj.libName,'simxAuxiliaryConsoleShow',clientID,consoleHandle_,showState_,operation
Mode_);
        end

        function rtn = simxBreakForceSensor(obj,clientID,forceSensorHandle,operationMode)
            forceSensorHandle_ = int32(forceSensorHandle);
            operationMode_ = int32(operationMode);

            rtn = calllib(obj.libName,
'simxBreakForceSensor',clientID,forceSensorHandle_,operationMode_);
        end

        function rtn = simxClearFloatSignal(obj,clientID,signalName,operationMode)
            signalName_ = libpointer('int8Ptr',[int8(signalName) 0]);
            operationMode_ = int32(operationMode);

            rtn = calllib(obj.libName,'simxClearFloatSignal',clientID,signalName_,operationMode_);
        end

        function rtn = simxClearIntegerSignal(obj,clientID,signalName,operationMode)
            signalName_ = libpointer('int8Ptr',[int8(signalName) 0]);
            operationMode_ = int32(operationMode);

            rtn =
calllib(obj.libName,'simxClearIntegerSignal',clientID,signalName_,operationMode_);
        end

        function rtn = simxClearStringSignal(obj,clientID,signalName,operationMode)
            signalName_ = libpointer('int8Ptr',[int8(signalName) 0]);
            operationMode_ = int32(operationMode);

            rtn = calllib(obj.libName,'simxClearStringSignal',clientID,signalName_,operationMode_);
        end

        function rtn = simxCloseScene(obj,clientID,operationMode)
            operationMode_ = int32(operationMode);
```

```
        rtn = calllib(obj.libName,'simxCloseScene',clientID,operationMode_);
    end

    function [rtn newObjectHandles] =
simxCopyPasteObjects(obj,clientID,objectHandles,operationMode)
        objectHandles_ = libpointer('int32Ptr',int32(objectHandles));
        objectCount_ = int32(numel(objectHandles));
        newObjectHandles_ = libpointer('int32Ptr',[]);
        newObjectCount_ = libpointer('int32Ptr',int32(0));
        operationMode_ = int32(operationMode);

        [rtn objectHandles_ newObjectHandles_ newObjectCount_] =
calllib(obj.libName,'simxCopyPasteObjects',clientID,objectHandles_,objectCount_,newObjectHa
ndles_,newObjectCount_,operationMode_);

        if (rtn==0)&&(newObjectCount_>0)
            newObjectHandles = zeros(1,newObjectCount_);
            newObjectHandles_.setdatatype('int32Ptr',1,newObjectCount_);
            for i=1:newObjectCount_;
                newObjectHandles(i) = newObjectHandles_.value(i);
            end
        else
            newObjectHandles=[];
        end
    end

    function buffer = simxCreateBuffer(obj,bufferSize)
        bufferSize_ = int32(bufferSize);

        buffer = calllib(obj.libName,'simxCreateBuffer',bufferSize_);
    end

    function rtn = simxEndDialog(obj,clientID,dialogHandle,operationMode)
        operationMode_ = int32(operationMode);
        dialogHandle_ = int32(dialogHandle);

        rtn = calllib(obj.libName,'simxEndDialog',clientID,dialogHandle_,operationMode_);
    end

    function rtn = simxEraseFile(obj,clientID,fileName_serverSide,operationMode)
        fileName_serverSide_ = libpointer('int8Ptr',[int8(fileName_serverSide) 0]);
        operationMode_ = int32(operationMode);
```

```
        [rtn fileName_serverSide_] =
calllib(obj.libName,'simxEraseFile',clientID,fileName_serverSide_,operationMode_);
    end

    function [rtn paramValues ]=
simxGetArrayParameter(obj,clientID,paramIdentifier,operationMode)
        paramIdentifier_ = int32(paramIdentifier);
        operationMode_ = int32(operationMode);
        paramValues = libpointer('singlePtr',single([0 0 0]));

        [rtn paramValues] =
calllib(obj.libName,'simxGetArrayParameter',clientID,paramIdentifier_,paramValues
,operationMode_);
    end

    function [rtn paramValues ]=
simxGetBooleanParameter(obj,clientID,paramIdentifier,operationMode)
        paramIdentifier_ = int32(paramIdentifier);
        operationMode_ = int32(operationMode);
        paramValues = libpointer('uint8Ptr',uint8(0));

        [rtn paramValues] =
calllib(obj.libName,'simxGetBooleanParameter',clientID,paramIdentifier_,paramValues
,operationMode_);
    end

    function [rtn handle] =
simxGetCollisionHandle(obj,clientID,collisionObjectName,operationMode)
        collisionObjectName_ = libpointer('int8Ptr',[int8(collisionObjectName) 0]);
        handle_ = libpointer('int32Ptr',int32(0));
        operationMode_ = int32(operationMode);

        [rtn collisionObjectName_ handle] =
calllib(obj.libName,'simxGetCollisionHandle',clientID,collisionObjectName_,handle_
,operationMode_);
    end

    function [rtn handle] =
simxGetCollectionHandle(obj,clientID,collectionName,operationMode)
        collectionName_ = libpointer('int8Ptr',[int8(collectionName) 0]);
        handle_ = libpointer('int32Ptr',int32(0));
        operationMode_ = int32(operationMode);
```

```
        [rtn collectionName_ handle] =
calllib(obj.libName,'simxGetCollectionHandle',clientID,collectionName_,handle_
,operationMode_);
    end

    function rtn = simxGetConnectionId(obj,clientID)
        rtn = calllib(obj.libName,'simxGetConnectionId',clientID);
    end

     function [rtn inputText]= simxGetDialogInput(obj,clientID,dialogHandle,operationMode)
        dialogHandle_ = int32(dialogHandle);
        inputText_ = libpointer('int8PtrPtr');
        operationMode_ = int32(operationMode);

        [rtn inputText_ ] =
calllib(obj.libName,'simxGetDialogInput',clientID,dialogHandle_,inputText_,operationMode_);

        if (rtn==0)
            s=1;
            inputText_.setdatatype('int8Ptr',1,s);
            value = inputText_.value(s);
            while(value ~= 0)
                inputText_.setdatatype('int8Ptr',1,s);
                value = inputText_.value(s);
                s=s+1;
            end
            tmp = inputText_.value(1:s-1);
            inputText = char(tmp);
        else
            inputText = [];
        end
    end

    function [rtn result]= simxGetDialogResult (obj,clientID,dialogHandle,operationMode)
        dialogHandle_  = int32(dialogHandle);
        result = libpointer('int32Ptr',int32(0));
        operationMode_ = int32(operationMode);

        [rtn result] =
calllib(obj.libName,'simxGetDialogResult',clientID,dialogHandle_,result,operationMode_);
    end
```

```
        function [rtn handle] =
simxGetDistanceHandle(obj,clientID,distanceObjectName,operationMode)
            distanceObjectName_ = libpointer('int8Ptr',[int8(distanceObjectName) 0]);
            handle_ = libpointer('int32Ptr',int32(0));
            operationMode_ = int32(operationMode);

            [rtn distanceObjectName_ handle] =
calllib(obj.libName,'simxGetDistanceHandle',clientID,distanceObjectName_,handle_
,operationMode_);
        end

        function [rtn paramValue]=
simxGetFloatingParameter(obj,clientID,paramIdentifier,operationMode)
            paramIdentifier_ = int32(paramIdentifier);
            operationMode_ = int32(operationMode);
            paramValue = libpointer('singlePtr',single(0));

            [rtn paramValue] =
calllib(obj.libName,'simxGetFloatingParameter',clientID,paramIdentifier_,paramValue,operation
Mode_);
        end

        function [rtn signalValue]= simxGetFloatSignal(obj,clientID,signalName,operationMode)
            signalName_ = libpointer('int8Ptr',[int8(signalName) 0]);
            signalValue = libpointer('singlePtr',single(0));
            operationMode_ = int32(operationMode);

            [rtn signalName_ signalValue] =
calllib(obj.libName,'simxGetFloatSignal',clientID,signalName_,signalValue,operationMode_);
        end

        function [rtn info]= simxGetInMessageInfo(obj,clientID,infoType)
            infoType_ = int32(infoType);
            info = libpointer('int32Ptr',int32(0));

            [rtn info] = calllib(obj.libName,'simxGetInMessageInfo',clientID,infoType_,info);
        end

        function [rtn paramValue]= simxGetIntegerParameter
(obj,clientID,paramIdentifier,operationMode)
            paramIdentifier_ = int32(paramIdentifier);
            operationMode_ = int32(operationMode);
            paramValue = libpointer('int32Ptr',int32(0));
```

```
        [rtn paramValue] =
calllib(obj.libName,'simxGetIntegerParameter',clientID,paramIdentifier_,paramValue,operationM
ode_);
    end

    function [rtn signalValue]= simxGetIntegerSignal(obj,clientID,signalName,operationMode)
        signalName_ = libpointer('int8Ptr',[int8(signalName) 0]);
        signalValue = libpointer('int32Ptr',int32(0));
        operationMode_ = int32(operationMode);

        [rtn signalName signalValue] =
calllib(obj.libName,'simxGetIntegerSignal',clientID,signalName_,signalValue,operationMode_);
    end

    function [rtn matrix] = simxGetJointMatrix(obj,clientID,jointHandle,operationMode)
        jointHandle_ = int32(jointHandle);
        matrix = libpointer('singlePtr',single([0 0 0 0 0 0 0 0 0 0 0 0]));
        operationMode_ = int32(operationMode);

       [rtn matrix ] = calllib(obj.libName,'simxGetJointMatrix',clientID,jointHandle_,matrix,
operationMode_);
    end

    function [rtn position] = simxGetJointPosition(obj,clientID,handle,option)
        handle_ = int32(handle);
        option_ = int32(option);

        [rtn position] =
calllib(obj.libName,'simxGetJointPosition',clientID,handle_,single(43),option_);
    end

    function rtn = simxGetLastCmdTime(obj,clientID)
        rtn = calllib(obj.libName,'simxGetLastCmdTime',clientID);
    end

    function [rtn errorStrings]= simxGetLastErrors(obj,clientID,operationMode)
        errorCnt = libpointer('int32Ptr',int32(0));
        errorStrings_ = libpointer('int8PtrPtr');
        operationMode_ = int32(operationMode);

        [rtn errorCnt errorStrings_ ] =
calllib(obj.libName,'simxGetLastErrors',clientID,errorCnt,errorStrings_,operationMode_);
```

```
        if (rtn==0)
            errorStrings = cell(double(errorCnt));
            s=1;
            for i=1:errorCnt
                begin = s;
                errorStrings_.setdatatype('int8Ptr',1,s);
                value = errorStrings_.value(s);
                while(value ~= 0)
                    errorStrings_.setdatatype('int8Ptr',1,s);
                    value = errorStrings_.value(s);
                    s=s+1;
                end
                tmp = errorStrings_.value(begin:s-1);
                errorStrings(i) = cellstr(char(tmp));
            end
        else
            errorStrings=[];
        end
    end

    function [rtn prop] = simxGetModelProperty(obj,clientID,objectHandle,operationMode)
        objectHandle_ = int32(objectHandle);
        operationMode_ = int32(operationMode);
        prop = libpointer('int32Ptr',int32(0));

        [rtn prop ] =
calllib(obj.libName,'simxGetModelProperty',clientID,objectHandle_,prop,operationMode_);
    end

    function [rtn childObjectHandle] =
simxGetObjectChild(obj,clientID,parentObjectHandle,childIndex,operationMode)
        parentObjectHandle_ = int32(parentObjectHandle);
        childIndex_ = int32(childIndex);
        childObjectHandle = libpointer('int32Ptr',int32(0));
        operationMode_ = int32(operationMode);

        [rtn childObjectHandle ] =
calllib(obj.libName,'simxGetObjectChild',clientID,parentObjectHandle_,childIndex_,childObjectH
andle,operationMode_);
    end

    function [rtn paramValue]=
```

```
simxGetObjectFloatParameter(obj,clientID,objectHandle,parameterID,operationMode)
        objectHandle_ = int32(objectHandle);
        parameterID_ = int32(parameterID);
        operationMode_ = int32(operationMode);
        paramValue = libpointer('singlePtr',single(0));

        [rtn paramValue] =
calllib(obj.libName,'simxGetObjectFloatParameter',clientID,objectHandle_,parameterID_,param
Value,operationMode_);
    end

    function [rtn handle] = simxGetObjectHandle(obj,clientID,name,operationmode)
        name_ptr = libpointer('int8Ptr',[uint8(name) 0]);
        handle_ptr = libpointer('int32Ptr',int32(0));
        operationmode_ = int32(operationmode);

        [rtn name_ptr handle] =
calllib(obj.libName,'simxGetObjectHandle',clientID,name_ptr,handle_ptr,operationmode_);
    end

    function [rtn paramValue]=
simxGetObjectIntParameter(obj,clientID,objectHandle,parameterID,operationMode)
        objectHandle_ = int32(objectHandle);
        parameterID_ = int32(parameterID);
        operationMode_ = int32(operationMode);
        paramValue = libpointer('int32Ptr',int32(0));

        [rtn paramValue] =
calllib(obj.libName,'simxGetObjectIntParameter',clientID,objectHandle_,parameterID_,paramVal
ue,operationMode_);
    end

    function [rtn eulerAngles] =
simxGetObjectOrientation(obj,clientID,objectHandle,relativeToObjectHandle,operationMode)
        objectHandle_ = int32(objectHandle);
        relativeToObjectHandle_ = int32(relativeToObjectHandle);
        operationMode_ = int32(operationMode);
        eulerAngles = libpointer('singlePtr', single([0 0 0]));

        [rtn eulerAngles] =
calllib(obj.libName,'simxGetObjectOrientation',clientID,objectHandle_,relativeToObjectHandle_,
eulerAngles ,operationMode_);
    end
```

```
        function [rtn quaternion_coeffs] =
simxGetObjectQuaternion(obj,clientID,objectHandle,relativeToObjectHandle,operationMode)
            objectHandle_ = int32(objectHandle);
            relativeToObjectHandle_ = int32(relativeToObjectHandle);
            operationMode_ = int32(operationMode);
            quaternion_coeffs = libpointer('singlePtr', single([0 0 0 0]));

            [rtn quaternion_coeffs] =
calllib(obj.libName,'simxGetObjectQuaternion',clientID,objectHandle_,relativeToObjectHandle_,
quaternion_coeffs ,operationMode_);
        end

        function [rtn parentObjectHandle] =
simxGetObjectParent(obj,clientID,objectHandle,operationMode)
            objectHandle_ = int32(objectHandle);
            parentObjectHandle = libpointer('int32Ptr',int32(0));
            operationMode_ = int32(operationMode);

            [rtn parentObjectHandle] =
calllib(obj.libName,'simxGetObjectParent',clientID,objectHandle_,parentObjectHandle,operation
Mode);
        end

        function [rtn position] =
simxGetObjectPosition(obj,clientID,objectHandle,relativeToObjectHandle,operationMode)
            objectHandle_ = int32(objectHandle);
            relativeToObjectHandle_ = int32(relativeToObjectHandle);
            operationMode_ = int32(operationMode);
            position = libpointer('singlePtr', single([0 0 0]));

            [rtn position] =
calllib(obj.libName,'simxGetObjectPosition',clientID,objectHandle_,relativeToObjectHandle_,posi
tion ,operationMode_);
        end

        function [rtn objectHandles] = simxGetObjects(obj,clientID,objectType,operationMode)
            objectType_ = int32(objectType);
            objectHandles_ = libpointer('int32PtrPtr');
            objectCount = libpointer('int32Ptr',int32(0));
            operationMode_ = int32(operationMode);

            [rtn objectCount objectHandles_]  =
```

```
calllib(obj.libName,'simxGetObjects',clientID,objectType_,objectCount,objectHandles_,operation
Mode_);

        if (rtn==0)
            if(objectCount > 0 )
                objectHandles_.setdatatype('int32Ptr',1,objectCount);
                objectHandles = objectHandles_.value;
            else
                objectHandles = [];
            end
        else
            objectHandles = [];
        end
    end

    function [rtn objectHandles ] = simxGetObjectSelection(obj,clientID,operationMode)
        objectHandles_ = libpointer('int32PtrPtr');
        objectCount = libpointer('int32Ptr',int32(0));
        operationMode_ = int32(operationMode);

        [rtn objectHandles_ objectCount] =
calllib(obj.libName,'simxGetObjectSelection',clientID,objectHandles_
,objectCount,operationMode_);

        if (rtn==0)
            if(objectCount > 0)
                objectHandles_.setdatatype('int32Ptr',1,objectCount);
                objectHandles = objectHandles_.value;
            else
                objectHandles = [];
            end
        else
            objectHandles = [];
        end
    end

    function [rtn info]= simxGetOutMessageInfo(obj,clientID,infoType)
        infoType_ = int32(infoType);
        info = libpointer('int32Ptr',int32(0));

        [rtn info] = calllib(obj.libName,'simxGetOutMessageInfo',clientID,infoType_,info);
    end
```

```
        function [rtn pingTime]= simxGetPingTime(obj,clientID)
            pingTime = libpointer('int32Ptr',int32(0));

            [rtn pingTime] = calllib(obj.libName,'simxGetPingTime',clientID,pingTime);
        end

        function [rtn paramValue]=
simxGetStringParameter(obj,clientID,paramIdentifier,operationMode)
            paramIdentifier_ = int32(paramIdentifier);
            operationMode_ = int32(operationMode);
            paramValue_ = libpointer('int8PtrPtr');

            [rtn paramValue_] =
calllib(obj.libName,'simxGetStringParameter',clientID,paramIdentifier_,paramValue_,operationM
ode_);

            if(rtn == 0)
                s=1;
                paramValue_.setdatatype('int8Ptr',1,s);
                value = paramValue_.value(s);
                while(value ~= 0)
                    paramValue_.setdatatype('int8Ptr',1,s);
                    value = paramValue_.value(s);
                    s=s+1;
                end
                tmp = paramValue_.value(1:s-1);
                paramValue = char(tmp);
            else
                paramValue = [];
            end
        end

        function [rtn signalValue ]= simxGetStringSignal(obj,clientID,signalName,operationMode)
            signalName_ = libpointer('int8Ptr',[int8(signalName) 0]);
            signalValue_ = libpointer('uint8PtrPtr');
            signalLength= libpointer('int32Ptr',int32(0));
            operationMode_ = int32(operationMode);

            [rtn signalName_ signalValue_ signalLength ] =
calllib(obj.libName,'simxGetStringSignal',clientID,signalName_,signalValue_,signalLength
,operationMode_);

            if (rtn==0)
```

```
            if (signalLength>0)
                signalValue_.setdatatype('uint8Ptr',1,double(signalLength));
                signalValue = char(signalValue_.value);
            else
                signalValue = [];
            end
        else
            signalValue = [];
        end
    end

    function [rtn signalValue ]=
simxGetAndClearStringSignal(obj,clientID,signalName,operationMode)
        signalName_ = libpointer('int8Ptr',[int8(signalName) 0]);
        signalValue_ = libpointer('uint8PtrPtr');
        signalLength= libpointer('int32Ptr',int32(0));
        operationMode_ = int32(operationMode);

        [rtn signalName_ signalValue_ signalLength ] =
calllib(obj.libName,'simxGetAndClearStringSignal',clientID,signalName_,signalValue_,signalLen
gth ,operationMode_);

        if (rtn==0)
            if (signalLength>0)
                signalValue_.setdatatype('uint8Ptr',1,double(signalLength));
                signalValue = char(signalValue_.value);
            else
                signalValue = [];
            end
        else
            signalValue = [];
        end
    end

    function [rtn prop] =
simxGetUIButtonProperty(obj,clientID,uiHandle,uiButtonID,operationMode)
        uiHandle_ = int32(uiHandle);
        operationMode_ = int32(operationMode);
        uiButtonID_ = int32(uiButtonID);
        prop = libpointer('int32Ptr',int32(0));

        [rtn prop ] =
calllib(obj.libName,'simxGetUIButtonProperty',clientID,uiHandle_,uiButtonID_,prop,operationMo
```

```
de_);
        end

        function [rtn uiEventButtonID auxValues] =
simxGetUIEventButton(obj,clientID,uiHandle,operationMode)
            uiHandle_ = int32(uiHandle);
            operationMode_ = int32(operationMode);
            uiEventButtonID = libpointer('int32Ptr',int32(0));
            auxValues = libpointer('int32Ptr',int32([0 0]));

            [rtn uiEventButtonID auxValues] =
calllib(obj.libName,'simxGetUIEventButton',clientID,uiHandle_,uiEventButtonID ,auxValues
,operationMode_);
        end

        function [rtn handle] = simxGetUIHandle(obj,clientID,uiName,operationMode)
            uiName_ = libpointer('int8Ptr',[int8(uiName) 0]);
            operationMode_ = int32(operationMode);
            handle = libpointer('int32Ptr',int32(0));

            [rtn uiName handle] =
calllib(obj.libName,'simxGetUIHandle',clientID,uiName_,handle,operationMode_);
        end

        function [rtn position] = simxGetUISlider(obj,clientID,uiHandle,uiButtonID,operationMode)
            uiHandle_ = int32(uiHandle);
            operationMode_ = int32(operationMode);
            uiButtonID_ = int32(uiButtonID);
            position = libpointer('int32Ptr',int32(0));

            [rtn position ] =
calllib(obj.libName,'simxGetUISlider',clientID,uiHandle_,uiButtonID_,position,operationMode_)
        end

        function [rtn resolution_ buffer] =
simxGetVisionSensorDepthBuffer(obj,clientID,handle,operationmode)
            resolution = [0 0];
            resolution_ = libpointer('int32Ptr',int32(resolution));
            buffer = libpointer('singlePtr',[]);
            operationmode_ = int32(operationmode);
            handle_ = int32(handle);

            [rtn resolution_ buffer] =
```

```
calllib(obj.libName,'simxGetVisionSensorDepthBuffer',clientID,handle_,resolution_,buffer,operationmode_);
        end

        function [rtn resolution_ buffer] = simxGetVisionSensorDepthBuffer2(obj,clientID,handle,operationmode)
            resolution = [0 0];
            resolution_ = libpointer('int32Ptr',int32(resolution));
            buffer_ = libpointer('singlePtr',[]);
            operationmode_ = int32(operationmode);
            handle_ = int32(handle);

            [rtn resolution_ buffer_] = calllib(obj.libName,'simxGetVisionSensorDepthBuffer',clientID,handle_,resolution_,buffer_,operationmode_);

            if (rtn==0)&(nargout>2)
                if((resolution_(1) ~= 0) && (resolution_(2) ~= 0))
                    buffer_.setdatatype('singlePtr',1,resolution_(1)*resolution_(2));
                    buffer = flipdim(permute(reshape(buffer_.Value, resolution_(1), resolution_(2)), [2 1]), 1);
                end
            else
                buffer = [];
            end
        end

%       function [rtn resolution_ buffer] = simxGetVisionSensorDepthBuffer2(obj,clientID,handle,operationmode)
%           resolution = [0 0];
%           resolution_ = libpointer('int32Ptr',int32(resolution));
%           buffer_ = libpointer('singlePtr',[]);
%           operationmode_ = int32(operationmode);
%           handle_ = int32(handle);
%
%           [rtn resolution_ buffer_] = calllib(obj.libName,'simxGetVisionSensorDepthBuffer',clientID,handle_,resolution_,buffer_,operationmode_);
%
%           if (rtn==0)
%               if((resolution_(1) ~= 0) && (resolution_(2) ~= 0))
%                   buffer_.setdatatype('singlePtr',1,resolution_(1)*resolution_(2));
%                   buffer = zeros(resolution_(1),resolution_(2));
```

```
%                 buffer = cast(buffer,'single');
%                 for i = resolution_(1):-1:1;
%                     count = (resolution_(1)-i)*resolution_(2);
%                     for j = 1:resolution_(2);
%                         buffer(i,j) = single(buffer_.value(count+j));
%                     end
%                 end
%             end
%         else
%             buffer = [];
%         end
%     end

    function [rtn resolution_ image] =
simxGetVisionSensorImage(obj,clientID,handle,options,operationmode)
        resolution = [0 0];
        resolution_ = libpointer('int32Ptr',int32(resolution));
        image = libpointer('uint8Ptr',[]);
        options_ = uint8(options);
        operationmode_ = int32(operationmode);
        handle_ = int32(handle);

        [rtn resolution_ image] =
calllib(obj.libName,'simxGetVisionSensorImage',clientID,handle_,resolution_,image,options_,op
erationmode_);
    end

    function [rtn resolution_ image] =
simxGetVisionSensorImage2(obj,clientID,handle,options,operationmode)
        resolution = [0 0];
        resolution_ = libpointer('int32Ptr',int32(resolution));
        image_ = libpointer('uint8Ptr',[]);
        options_ = uint8(options);
        operationmode_ = int32(operationmode);
        handle_ = int32(handle);

        [rtn resolution_ image_] =
calllib(obj.libName,'simxGetVisionSensorImage',clientID,handle_,resolution_,image_,options_,o
perationmode_);

        if (rtn==0)&(nargout>2)
            if((resolution_(1) ~= 0) && (resolution_(2) ~= 0))
                if(options == 1) %grayscale image
```

```
                image_.setdatatype('uint8Ptr',1,resolution_(1)*resolution_(2));
                image = flipdim(permute(reshape(image_.Value, resolution_(1), resolution_(2)),
[2 1]), 1);
            else
                image_.setdatatype('uint8Ptr',1,resolution_(1)*resolution_(2)*3);
                image = flipdim(permute(reshape(image_.Value, 3, resolution_(1),
resolution_(2)), [3 2 1]), 1);
            end
        end
    else
        image = [];
    end
end

function [rtn force]= simxJointGetForce(obj,clientID,jointHandle,operationMode)
    jointHandle_ = int32(jointHandle);
    operationMode_ = int32(operationMode);
    force = libpointer('single',single(0));

    [rtn force ] =
calllib(obj.libName,'simxGetJointForce',clientID,jointHandle_,single(0),operationMode_);
end

function [rtn force]= simxGetJointForce(obj,clientID,jointHandle,operationMode)
    jointHandle_ = int32(jointHandle);
    operationMode_ = int32(operationMode);
    force = libpointer('single',single(0));

    [rtn force ] =
calllib(obj.libName,'simxGetJointForce',clientID,jointHandle_,single(0),operationMode_);
end

function [rtn force]= simxGetJointMaxForce(obj,clientID,jointHandle,operationMode)
    jointHandle_ = int32(jointHandle);
    operationMode_ = int32(operationMode);
    force = libpointer('single',single(0));

    [rtn force ] =
calllib(obj.libName,'simxGetJointMaxForce',clientID,jointHandle_,single(0),operationMode_);
end

function [rtn baseHandle]=
simxLoadModel(obj,clientID,modelPathAndName,options,operationMode)
```

```
        modelPathAndName_ = libpointer('int8Ptr',[int8(modelPathAndName) 0]);
        options_ = uint8(options);
        baseHandle= libpointer('int32Ptr',int32(0));
        operationMode_ = int32(operationMode);

        [rtn modelPathAndName_ baseHandle] =
calllib(obj.libName,'simxLoadModel',clientID,modelPathAndName_,options_,baseHandle,operati
onMode_);
    end

    function [rtn ]= simxLoadScene(obj,clientID,scenePathAndName,options,operationMode)
        scenePathAndName_ = libpointer('int8Ptr',[int8(scenePathAndName) 0]);
        options_ = uint8(options);
        operationMode_ = int32(operationMode);

        [rtn scenePathAndName_ ] =
calllib(obj.libName,'simxLoadScene',clientID,scenePathAndName_,options_,operationMode_);
    end

    function [rtn uiHandles] = simxLoadUI(obj,clientID,uiPathAndName,options,operationMode)
        uiPathAndName_ = libpointer('int8Ptr',int8([uiPathAndName 0]));
        options_ = uint8(options);
        count = libpointer('int32Ptr', int32(0));
        uiHandles_ = libpointer('int32Ptr');
        operationMode_ = int32(operationMode);

        [rtn uiPathAndName_ count uiHandles_ ] =
calllib(obj.libName,'simxLoadUI',clientID,uiPathAndName_,options_,count ,uiHandles_
,operationMode_);

        if (rtn==0)
            if(count > 0)
                uiHandles_.setdatatype('int32Ptr',1,count);
                uiHandles = uiHandles_.value;
            else
                uiHandles = [];
            end
            uiHandles_.setdatatype('uint8Ptr',2,2);
            obj.simxReleaseBuffer(uiHandles_);
        else
            uiHandles = [];
        end
    end
```

```
        function rtn = simxPauseSimulation(obj,clientID,operationMode)
            operationMode_ = int32(operationMode);

            rtn = calllib(obj.libName,'simxPauseSimulation',clientID,operationMode_);
        end

        function [rtn collisionState]=
simxReadCollision(obj,clientID,collisionObjectHandle,operationMode)
            collisionObjectHandle_ = int32(collisionObjectHandle);
            operationMode_ = int32(operationMode);
            collisionState = libpointer('uint8Ptr',uint8(0));

            [rtn collisionState] =
calllib(obj.libName,'simxReadCollision',clientID,collisionObjectHandle_,collisionState,operationM
ode_);
        end

        function [rtn minimumDistance]=
simxReadDistance(obj,clientID,distanceObjectHandle,operationMode)
            distanceObjectHandle_ = int32(distanceObjectHandle);
            operationMode_ = int32(operationMode);
            minimumDistance = libpointer('singlePtr',single(0));

            [rtn minimumDistance] =
calllib(obj.libName,'simxReadDistance',clientID,distanceObjectHandle_,minimumDistance,opera
tionMode_);
        end

        function [rtn state forceVector torqueVector]=
simxReadForceSensor(obj,clientID,forceSensorHandle,operationMode)
            forceSensorHandle_ = int32(forceSensorHandle);
            state = libpointer('uint8Ptr', uint8(0));
            forceVector = libpointer('singlePtr', single([0 0 0]));
            torqueVector = libpointer('singlePtr', single([0 0 0]));
            operationMode_ = int32(operationMode);

            [rtn state forceVector torqueVector] =
calllib(obj.libName,'simxReadForceSensor',clientID,forceSensorHandle_,state ,forceVector,
torqueVector,operationMode_);
        end

        function [rtn detectionState auxValues auxValuesCount ] =
```

```
simxReadVisionSensor(obj,clientID,sensorHandle,operationmode)
        detectionState = libpointer('uint8Ptr',uint8(0));
        auxValues_ = libpointer('singlePtrPtr');
        auxValuesCount_ = libpointer('int32PtrPtr');
        operationmode_ = int32(operationmode);
        sensorHandle_ = int32(sensorHandle);

        [rtn detectionState auxValues_ auxValuesCount_] =
calllib(obj.libName,'simxReadVisionSensor',clientID,sensorHandle_,detectionState
,auxValues_,auxValuesCount_,operationmode_);

        auxValues = [];
        auxValuesCount = [];
    if ~isempty(auxValuesCount_),
      if isempty(auxValues_),
       error('Error: auxValues_ is empty (error 0). Please contact Renaud Detry and report
exactly this error message.');
      end
      if rtn == 0,
          auxValuesCount_.setdatatype('int32Ptr',1);
          packets=auxValuesCount_.value(1);
          auxValuesCount_.setdatatype('int32Ptr',packets+1);
          auxValuesCount = auxValuesCount_.Value(2:(packets+1));
          auxValues_.setdatatype('singlePtr',sum(auxValuesCount));
          if(rtn == 0)&(nargout>1)
            auxValues = auxValues_.Value';
          end
        end
          auxValuesCount_.setdatatype('uint8Ptr',2,2);
          auxValues_.setdatatype('uint8Ptr',2,2);
          obj.simxReleaseBuffer(auxValuesCount_);
          obj.simxReleaseBuffer(auxValues_);
        else
      if ~isempty(auxValues_),
       error('Error: auxValues_ is *not* empty (error 1). Please contact Renaud Detry and report
exactly this error message.');
      end
        end
      end

      function [] = simxReleaseBuffer(obj,buffer)
        buffer_ = calllib(obj.libName,'simxReleaseBuffer',buffer);
      end
```

```
        function [rtn ] = simxRemoveObject(obj,clientID,objectHandle,operationMode)
            objectHandle_ = int32(objectHandle);
            operationMode_ = int32(operationMode);

            [rtn ]  =
calllib(obj.libName,'simxRemoveObject',clientID,objectHandle_,operationMode_);
        end

        function [rtn ] = simxRemoveModel(obj,clientID,objectHandle,operationMode)
            objectHandle_ = int32(objectHandle);
            operationMode_ = int32(operationMode);

            [rtn ]  = calllib(obj.libName,'simxRemoveModel',clientID,objectHandle_,operationMode_);
        end

        function [rtn ] = simxRemoveUI(obj,clientID,uiHandle,operationMode)
            uiHandle_ = int32(uiHandle);
            operationMode_ = int32(operationMode);

            [rtn ]  = calllib(obj.libName,'simxRemoveUI',clientID,uiHandle_,operationMode_);
        end

        function [rtn ]=
simxSetArrayParameter(obj,clientID,paramIdentifier,paramValues,operationMode)
            paramIdentifier_ = int32(paramIdentifier);
            num_ele = numel(paramValues);
            if (num_ele < 3)
                error('paramValues should have 3 values');
            else
                paramValues_ = libpointer('singlePtr',single(paramValues));
                operationMode_ = int32(operationMode);

                [rtn paramValues_ ] =
calllib(obj.libName,'simxSetArrayParameter',clientID,paramIdentifier_,paramValues_,operationM
ode_);
            end
        end

        function [rtn ]=
simxSetBooleanParameter(obj,clientID,paramIdentifier,paramValue,operationMode)
            paramIdentifier_ = int32(paramIdentifier);
            paramValue_ = uint8(paramValue);
```

```
        operationMode_ = int32(operationMode);

        [rtn ] =
calllib(obj.libName,'simxSetBooleanParameter',clientID,paramIdentifier_,paramValue_,operation
Mode_);
    end

    function [rtn ]=
simxSetIntegerParameter(obj,clientID,paramIdentifier,paramValue,operationMode)
        paramIdentifier_ = int32(paramIdentifier);
        paramValue_ = int32(paramValue);
        operationMode_ = int32(operationMode);

        [rtn ] =
calllib(obj.libName,'simxSetIntegerParameter',clientID,paramIdentifier_,paramValue_,operation
Mode_);
    end

    function [rtn ]= simxSetIntegerSignal(obj,clientID,signalName,signalValue,operationMode)
        signalName_ = libpointer('int8Ptr',[int8(signalName) 0]);
        signalValue_ = int32(signalValue);
        operationMode_ = int32(operationMode);

        [rtn signalName_ ] =
calllib(obj.libName,'simxSetIntegerSignal',clientID,signalName_,signalValue_,operationMode_);
    end

    function [rtn ]= simxSetModelProperty(obj,clientID,objectHandle,prop,operationMode)
        objectHandle_ = int32(objectHandle);
        prop_ = int32(prop);
        operationMode_ = int32(operationMode);

        [rtn ] =
calllib(obj.libName,'simxSetModelProperty',clientID,objectHandle_,prop_,operationMode_);
    end

    function [rtn ]=
simxSetObjectIntParameter(obj,clientID,objectHandle,parameterID,parameterValue,operationM
ode)
        objectHandle_ = int32(objectHandle);
        parameterID_ = int32(parameterID);
        parameterValue_ = int32(parameterValue);
        operationMode_ = int32(operationMode);
```

```
        [rtn ] =
calllib(obj.libName,'simxSetObjectIntParameter',clientID,objectHandle_,parameterID_,parameter
Value_,operationMode_);
    end

    function [rtn ]=
simxSetObjectOrientation(obj,clientID,objectHandle,relativeToObjectHandle,eulerAngles,operati
onMode)
        objectHandle_ = int32(objectHandle);
        relativeToObjectHandle_ = int32(relativeToObjectHandle);
        num_ele = numel(eulerAngles);
        if (num_ele < 3)
            error('Euler angles should have 3 values');
            return;
        else
        eulerAngles_ = libpointer('singlePtr',single(eulerAngles));
        operationMode_ = int32(operationMode);

        [rtn ] =
calllib(obj.libName,'simxSetObjectOrientation',clientID,objectHandle_,relativeToObjectHandle_,e
ulerAngles_,operationMode_);
        end
    end

    function [rtn ]=
simxSetObjectQuaternion(obj,clientID,objectHandle,relativeToObjectHandle,quaternion_coeffs,
operationMode)
        objectHandle_ = int32(objectHandle);
        relativeToObjectHandle_ = int32(relativeToObjectHandle);
        num_ele = numel(quaternion_coeffs);
        if (num_ele < 4)
            error('A quaternion should have 4 values');
            return;
        else
        quaternion_ = libpointer('singlePtr',single(quaternion_coeffs));
        operationMode_ = int32(operationMode);

        [rtn ] =
calllib(obj.libName,'simxSetObjectQuaternion',clientID,objectHandle_,relativeToObjectHandle_,q
uaternion_,operationMode_);
        end
    end
```

```
        function [rtn] =
simxSetObjectParent(obj,clientID,objectHandle,parentObject,keepInPlace,operationMode)
            objectHandle_ = int32(objectHandle);
            parentObject_ = int32(parentObject);
            keepInPlace_ = uint8(keepInPlace);
            operationMode_ = int32(operationMode);

            [rtn ] = calllib(obj.libName,'simxSetObjectParent',clientID,objectHandle_
,parentObject_,keepInPlace_,operationMode_);
        end

        function rtn = simxSetObjectPosition(obj,clientID,handle,rel_pos,position,option)
            if (numel(position) < 3)
                error('position should have 3 values');
                return;
            end

            handle_ = int32(handle);
            rel_pos_ = int32(rel_pos);
            option_ = int32(option);
            pos_ptr = libpointer('singlePtr',single(position));

           [rtn pos_ptr] =
calllib(obj.libName,'simxSetObjectPosition',clientID,handle_,rel_pos_,pos_ptr,option_);
        end

        function [rtn] = simxSetObjectSelection(obj,clientID,objectHandles,operationMode)
            objectHandles_ = libpointer('int32Ptr',int32(objectHandles));
            objectCount = numel(objectHandles);
            operationMode_ = int32(operationMode);

            [rtn objectHandles_ ] =
calllib(obj.libName,'simxSetObjectSelection',clientID,objectHandles_
,objectCount,operationMode_);
        end

        function [rtn ]= simxSetSphericalJointMatrix(obj,clientID,jointHandle,matrix,operationMode)
            jointHandle_ = int32(jointHandle);
            num_ele = numel(matrix);
            if (num_ele < 12)
                error('matrix should have 12 values');
                return;
```

```
        else
            matrix_ = libpointer('singlePtr',single(matrix));
            operationMode_ = int32(operationMode);

            [rtn signalName_ ] =
calllib(obj.libName,'simxSetSphericalJointMatrix',clientID,jointHandle_,matrix_,operationMode_);
        end
    end

    function [rtn ]= simxSetStringSignal(obj,clientID,signalName,signalValue,operationMode)
        signalLength_ = int32(length(signalValue));
        signalName_ = libpointer('int8Ptr',[int8(signalName) 0]);
        signalValue_ = libpointer('uint8Ptr',[uint8(signalValue) 0]);
        operationMode_ = int32(operationMode);

        [rtn signalName_ signalValue_ ] =
calllib(obj.libName,'simxSetStringSignal',clientID,signalName_,signalValue_,signalLength_
,operationMode_);
    end

    function [rtn ]=
simxAppendStringSignal(obj,clientID,signalName,signalValue,operationMode)
        signalLength_ = int32(length(signalValue));
        signalName_ = libpointer('int8Ptr',[int8(signalName) 0]);
        signalValue_ = libpointer('uint8Ptr',[uint8(signalValue) 0]);
        operationMode_ = int32(operationMode);

        [rtn signalName_ signalValue_ ] =
calllib(obj.libName,'simxAppendStringSignal',clientID,signalName_,signalValue_,signalLength_
,operationMode_);
    end

    function [rtn ] =
simxSetUIButtonLabel(obj,clientID,uiHandle,uiButtonID,upStateLabel,downStateLabel,operation
Mode)
        uiHandle_ = int32(uiHandle);
        operationMode_ = int32(operationMode);
        uiButtonID_ = int32(uiButtonID);
        upStateLabel_ = libpointer('int8Ptr',int8([upStateLabel 0]));
        downStateLabel_ = libpointer('int8Ptr',int8([downStateLabel 0]));

        [rtn upStateLabel_ downStateLabel_] =
calllib(obj.libName,'simxSetUIButtonLabel',clientID,uiHandle_,uiButtonID_,upStateLabel_,down
```

```
StateLabel_,operationMode_);
    end

    function [rtn ] =
simxSetUIButtonProperty(obj,clientID,uiHandle,uiButtonID,prop,operationMode)
        uiHandle_ = int32(uiHandle);
        operationMode_ = int32(operationMode);
        uiButtonID_ = int32(uiButtonID);
        prop_ = int32(prop);

        [rtn ] =
calllib(obj.libName,'simxSetUIButtonProperty',clientID,uiHandle_,uiButtonID_,prop_,operationMo
de_);
    end

    function [rtn] = simxSetUISlider (obj,clientID,uiHandle,uiButtonID,position,operationMode)
        uiHandle_ = int32(uiHandle);
        operationMode_ = int32(operationMode);
        uiButtonID_ = int32(uiButtonID);
        position_ = int32(position);

        [rtn] =
calllib(obj.libName,'simxSetUISlider',clientID,uiHandle_,uiButtonID_,position_,operationMode_)
    end

    function rtn =
simxSetVisionSensorImage(obj,clientID,handle,image,buffsize,options,operationmode)
        handle_ = int32(handle);
        buffsize_ = int32(buffsize);
        options_ = uint8(options);
        operationmode_ = int32(operationmode);

        [rtn image] =
calllib(obj.libName,'simxSetVisionSensorImage',clientID,handle_,image,buffsize_,options_,oper
ationmode_);
    end

    function rtn = simxSetVisionSensorImage2(obj,clientID,handle,image,operationmode)
        handle_ = int32(handle);
        [m n o] = size(image);
        buffsize_ = int32(m*n*o);
        if(o == 1)
            options_ = uint8(1);
```

```
        else
            options_ = uint8(0);
        end
        operationmode_ = int32(operationmode);

        % optimization courtesy of Renaud Detry:
        imdata= cast(reshape(permute(flipdim(image, 1), [3 2 1]), 1, buffsize_),'uint8');

%         imdata = zeros(1,buffsize_);
%         imdata= cast(imdata,'uint8');
%
%         for i = m:-1:1;
%             count = (m-i)*n*o;
%             for j = 1:o:n*o;
%                 for k=1:o;
%                     if(o==1)
%                         l=j;
%                     else
%                         l=int32(j/o) +1;
%                     end
%                     imdata(count+j+k-1) = image(i,l,k);
%                 end
%             end
%         end

        image_ = libpointer('uint8Ptr',imdata);

        [rtn image_] =
calllib(obj.libName,'simxSetVisionSensorImage',clientID,handle_,image_,buffsize_,options_,ope
rationmode_);
    end

    function rtn = simxStartSimulation(obj,clientID,operationMode)
        operationMode_ = int32(operationMode);

        rtn = calllib(obj.libName,'simxStartSimulation',clientID,operationMode_);
    end

    function rtn = simxStopSimulation(obj,clientID,operationMode)
        operationMode_ = int32(operationMode);

        rtn = calllib(obj.libName,'simxStopSimulation',clientID,operationMode_);
    end
```

```
        function rtn = simxSynchronous(obj,clientID,enable)
            enable_ = uint8(enable);

            rtn = calllib(obj.libName,'simxSynchronous',clientID,enable_);
        end

        function rtn = simxPauseCommunication(obj,clientID,enable)
            enable_ = uint8(enable);

            rtn = calllib(obj.libName,'simxPauseCommunication',clientID,enable_);
        end

        function rtn = simxSynchronousTrigger(obj,clientID)
            rtn = calllib(obj.libName,'simxSynchronousTrigger',clientID);
        end

        function rtn =
simxTransferFile(obj,clientID,filePathAndName,fileName_serverSide,timeOut,operationMode)
            filePathAndName_ = libpointer('int8Ptr',[int8(filePathAndName) 0]);
            fileName_serverSide_ = libpointer('uint8Ptr',[uint8(fileName_serverSide) 0]);
            timeOut_ = int32(timeOut);
            operationMode_ = int32(operationMode);

            [rtn filePathAndName_ fileName_serverSide_] =
calllib(obj.libName,'simxTransferFile',clientID,filePathAndName_,fileName_serverSide_,timeOut
_,operationMode_);
        end

        function [rtn linearVelocity angularVelocity]=
simxGetObjectVelocity(obj,clientID,objectHandle,operationMode)
            objectHandle_ = int32(objectHandle);
            linearVelocity = libpointer('singlePtr', single([0 0 0]));
            angularVelocity = libpointer('singlePtr', single([0 0 0]));
            operationMode_ = int32(operationMode);

            [rtn linearVelocity angularVelocity] =
calllib(obj.libName,'simxGetObjectVelocity',clientID,objectHandle_,linearVelocity,
angularVelocity,operationMode_);
        end

        function [string]= simxPackInts(obj,intArray)
            string=char(typecast(int32(intArray),'uint8'));
```

```
        end
        function [intArray]= simxUnpackInts(obj,string)
            intArray=typecast(uint8(int32(string)),'int32');
        end
        function [string]= simxPackFloats(obj,floatArray)
            string=char(typecast(single(floatArray),'uint8'));
        end
        function [floatArray]= simxUnpackFloats(obj,string)
            floatArray=typecast(uint8(single(string)),'single');
        end









        function rtn = simxSetJointPosition(obj,clientID,handle,position,option)
            handle_ = int32(handle);
            position_ = libpointer('singlePtr',single(position));
            option_ = int32(option);
            rtn = calllib(obj.libName,'mtlb_simxSetJointPosition',clientID,handle_,position_,option_);
        end

        function [rtn ]=
simxSetJointTargetVelocity(obj,clientID,objectHandle,targetVelocity,operationMode)
            objectHandle_ = int32(objectHandle);
            targetVelocity_ = libpointer('singlePtr',single(targetVelocity));
            operationMode_ = int32(operationMode);

            [rtn ] =
calllib(obj.libName,'mtlb_simxSetJointTargetVelocity',clientID,objectHandle_,targetVelocity_,ope
rationMode_);
        end

        function [rtn ]=
simxSetJointTargetPosition(obj,clientID,objectHandle,targetPosition,operationMode)
            objectHandle_ = int32(objectHandle);
            targetPosition_ = libpointer('singlePtr',single(targetPosition));
            operationMode_ = int32(operationMode);

            [rtn ] =
calllib(obj.libName,'mtlb_simxSetJointTargetPosition',clientID,objectHandle_,targetPosition_,ope
```

```
rationMode_);
    end

    function [rtn ]= simxSetJointForce(obj,clientID,objectHandle,force,operationMode)
        objectHandle_ = int32(objectHandle);
        force_ = libpointer('singlePtr',single(force));
        operationMode_ = int32(operationMode);

        [rtn ] =
calllib(obj.libName,'mtlb_simxSetJointForce',clientID,objectHandle_,force_,operationMode_);
    end

    function [rtn ]= simxSetJointMaxForce(obj,clientID,objectHandle,force,operationMode)
        objectHandle_ = int32(objectHandle);
        force_ = libpointer('singlePtr',single(force));
        operationMode_ = int32(operationMode);

        [rtn ] =
calllib(obj.libName,'mtlb_simxSetJointMaxForce',clientID,objectHandle_,force_,operationMode_)
;
    end

    function [rtn ]= simxSetFloatSignal(obj,clientID,signalName,signalValue,operationMode)
        signalName_ = libpointer('int8Ptr',[int8(signalName) 0]);
        signalValue_ = libpointer('singlePtr',single(signalValue));
        operationMode_ = int32(operationMode);

        [rtn signalName_ ] =
calllib(obj.libName,'mtlb_simxSetFloatSignal',clientID,signalName_,signalValue_,operationMode
_);
    end

    function [rtn ]=
simxSetObjectFloatParameter(obj,clientID,objectHandle,parameterID,parameterValue,operation
Mode)
        objectHandle_ = int32(objectHandle);
        parameterID_ = int32(parameterID);
        parameterValue_ = libpointer('singlePtr',single(parameterValue));
        operationMode_ = int32(operationMode);

        [rtn ] =
calllib(obj.libName,'mtlb_simxSetObjectFloatParameter',clientID,objectHandle_,parameterID_,p
arameterValue_,operationMode_);
```

```
        end

        function [rtn ]=
simxSetFloatingParameter(obj,clientID,paramIdentifier,paramValue,operationMode)
            paramIdentifier_ = int32(paramIdentifier);
            paramValue_ = libpointer('singlePtr',single(paramValue));
            operationMode_ = int32(operationMode);

            [rtn ] =
calllib(obj.libName,'mtlb_simxSetFloatingParameter',clientID,paramIdentifier_,paramValue_,ope
rationMode_);
        end

        function [rtn handle] = simxCreateDummy(obj,clientID,size,colors,operationmode)
            size_ = libpointer('singlePtr',single(size));
            operationmode_ = int32(operationmode);

            if (numel(colors) < 12)&&(numel(colors) ~= 0)
                error('colors should have 12 values');
                return;
            end

            color_ = libpointer('uint8Ptr',uint8(colors));
            handle_ = libpointer('int32Ptr',int32(0));

            [rtn s c handle] =
calllib(obj.libName,'mtlb_simxCreateDummy',clientID,size_,color_,handle_,operationmode_);
        end




        function [rtn detectionState detectedPoint detectedObjectHandle
detectedSurfaceNormalVector]=
simxReadProximitySensor(obj,clientID,sensorHandle,operationMode)
            clientIDandSensorHandle = libpointer('int32Ptr',int32([clientID,sensorHandle]));
            detectionState = libpointer('uint8Ptr', uint8(0));
            detectedPoint = libpointer('singlePtr', single([0 0 0]));
            detectedSurfaceNormalVector = libpointer('singlePtr', single([0 0 0]));
            detectedObjectHandle = libpointer('int32Ptr',int32(0));
            operationMode_ = int32(operationMode);

            [rtn clientIDandSensorHandle detectionState detectedPoint detectedObjectHandle
```

```
detectedSurfaceNormalVector] =
calllib(obj.libName,'mtlb_simxReadProximitySensor',clientIDandSensorHandle,detectionState
,detectedPoint , detectedObjectHandle ,detectedSurfaceNormalVector,operationMode_);
        end

        function [rtn console_handle] =
simxAuxiliaryConsoleOpen(obj,clientID,title,maxLines,mode,position,size,textcolor,backgroundc
olor,operationmode)

            posx=-10000;
            posy=-10000;
            sizex=-10000;
            sizey=-10000;

            if (numel(position) < 2)&&(numel(position) ~= 0)
                error('position should have 2 values');
                return;
            end
            if (numel(size) < 2)&&(numel(size) ~= 0)
                error('size should have 2 values');
                return;
            end

            if (numel(position) >= 2)
                posx=position(1);
                posy=position(2);
            end
            if (numel(size) >= 2)
                sizex=size(1);
                sizey=size(2);
            end

            clientIDandMaxLinesAndModeAndPositionAndSize =
libpointer('int32Ptr',int32([clientID,maxLines,mode,posx,posy,sizex,sizey]));
            title_ = libpointer('int8Ptr',[int8(title) 0]);

            if (numel(textcolor) < 3)&&(numel(textcolor) ~= 0)
                error('textcolor should have 3 values');
                return;
            end
            if (numel(backgroundcolor) < 3)&&(numel(backgroundcolor) ~= 0)
                error('backgroundcolor should have 3 values');
                return;
```

```
        end

        textcolor_ = libpointer('singlePtr',single(textcolor));
        backgroundcolor_ = libpointer('singlePtr',single(backgroundcolor));
        consolehandle_ = libpointer('int32Ptr',int32(0));
        operationmode_ = int32(operationmode);

        [rtn clientIDandMaxLinesAndModeAndPositionAndSize a b c console_handle] =
calllib(obj.libName,'mtlb_simxAuxiliaryConsoleOpen',clientIDandMaxLinesAndModeAndPosition
AndSize,title_,textcolor_,backgroundcolor_,consolehandle_,operationmode);
    end

    function [rtn dialogHandle uiHandle] =
simxDisplayDialog(obj,clientID,titleText,mainText,dialogType,initialText,titleColors,dialogColors,
operationMode)

        if (numel(titleColors) < 6)&&(numel(titleColors) ~= 0)
            error('titleColors should have 6 values');
            return;
        end
        if (numel(dialogColors) < 6)&&(numel(dialogColors) ~= 0)
            error('dialogColors should have 6 values');
            return;
        end

        clientIDandDlgTypeAndOperationMode =
libpointer('int32Ptr',int32([clientID,dialogType,operationMode]));
        colors = [-20000.0 -20000.0 -20000.0 -20000.0 -20000.0 -20000.0 -20000.0 -20000.0
-20000.0 -20000.0 -20000.0 -20000.0];
        if (numel(titleColors) >= 6)
            for i = 1:6
                colors(i)=titleColors(i)
            end
        end
        if (numel(dialogColors) >= 6)
            for i = 1:6
                colors(6+i)=dialogColors(i)
            end
        end

        clientHandleAndUiHandle = libpointer('int32Ptr',int32([0,0]));

        titleText_ = libpointer('int8Ptr',[int8(titleText) 0]);
```

```
        mainText_ = libpointer('int8Ptr',[int8(mainText) 0]);
        initialText_ = libpointer('int8Ptr',[int8(initialText) 0]);

        [rtn clientIDandDlgTypeAndOperationMode b c d colors clientHandleAndUiHandle] =
calllib(obj.libName,'mtlb_simxDisplayDialog',clientIDandDlgTypeAndOperationMode,titleText_,m
ainText_,initialText_,colors,clientHandleAndUiHandle);

        dialogHandle = clientHandleAndUiHandle(1);
        uiHandle = clientHandleAndUiHandle(2);
    end

    function [rtn retSignalValue]=
simxQuery(obj,clientID,signalName,signalValue,retSignalName,timeOutInMs)

        clientIDandSignalLengthAndTimeOutInMs =
libpointer('int32Ptr',int32([clientID,length(signalValue),timeOutInMs]));

        signalName_ = libpointer('int8Ptr',[int8(signalName) 0]);
        signalValue_ = libpointer('uint8Ptr',[uint8(signalValue) 0]);
        retSignalName_ = libpointer('int8Ptr',[int8(retSignalName) 0]);
        retSignalValue_ = libpointer('uint8PtrPtr');
        retSignalLength_= libpointer('int32Ptr',int32(0));
        timeOutInMs_ = int32(timeOutInMs);

        [rtn clientIDandSignalLengthAndTimeOutInMs signalName_,signalValue_,
retSignalName_, retSignalValue_, retSignalLength_] =
calllib(obj.libName,'mtlb_simxQuery',clientIDandSignalLengthAndTimeOutInMs,signalName_,sig
nalValue_, retSignalName_, retSignalValue_, retSignalLength_);

        if (rtn==0)
            if (retSignalLength_>0)
                retSignalValue_.setdatatype('uint8Ptr',1,double(retSignalLength_));
                retSignalValue = char(retSignalValue_.value);
            else
                retSignalValue = [];
            end
        else
            retSignalValue = [];
        end
    end

    function [rtn retHandles retInts retFloats retStrings]=
simxGetObjectGroupData(obj,clientID,objectType,dataType,operationMode)
```

```
        clientIDandObjectTypeAndDataTypeAndOperationMode =
libpointer('int32Ptr',int32([clientID,objectType,dataType,operationMode]));

        handlesCountAndIntDataCountAndFloatDataCountAndStringDataCount =
libpointer('int32Ptr',int32([0,0,0,0]));

        retHandles_ = libpointer('int32PtrPtr');
        retInts_ = libpointer('int32PtrPtr');
        retFloats_ = libpointer('singlePtrPtr');
        retStrings_ = libpointer('int8PtrPtr');

        [rtn clientIDandObjectTypeAndDataTypeAndOperationMode
handlesCountAndIntDataCountAndFloatDataCountAndStringDataCount retHandles_ retInts_
retFloats_ retStrings_ ] =
calllib(obj.libName,'mtlb_simxGetObjectGroupData',clientIDandObjectTypeAndDataTypeAndOp
erationMode,handlesCountAndIntDataCountAndFloatDataCountAndStringDataCount,retHandle
s_,retInts_,retFloats_,retStrings_);

        handlesCount_ =
handlesCountAndIntDataCountAndFloatDataCountAndStringDataCount(1);
        intsCount_ = handlesCountAndIntDataCountAndFloatDataCountAndStringDataCount(2);
        floatsCount_ =
handlesCountAndIntDataCountAndFloatDataCountAndStringDataCount(3);
        stringsCount_ =
handlesCountAndIntDataCountAndFloatDataCountAndStringDataCount(4);

        if (rtn==0)
            if (handlesCount_~=0)
                retHandles_.setdatatype('int32Ptr',1,handlesCount_);
                retHandles = retHandles_.value;
            else
                retHandles=[];
            end

            if (intsCount_~=0)
                retInts_.setdatatype('int32Ptr',1,intsCount_);
                retInts = retInts_.value;
            else
                retInts=[];
            end

            if (floatsCount_~=0)
```

```
            retFloats_.setdatatype('singlePtr',1,floatsCount_);
            retFloats = retFloats_.value;
        else
            retFloats=[];
        end

        retStrings = cell(double(stringsCount_));
        s=1;
        for i=1:stringsCount_
            begin = s;
            retStrings_.setdatatype('int8Ptr',1,s);
            value = retStrings_.value(s);
            while(value ~= 0)
                retStrings_.setdatatype('int8Ptr',1,s);
                value = retStrings_.value(s);
                s=s+1;
            end
            tmp = retStrings_.value(begin:s-1);
            retStrings(i) = cellstr(char(tmp));
        end

    else
        retHandles=[];
        retInts=[];
        retFloats=[];
        retStrings=[];
    end
end


function [rtn retInts retFloats retStrings retBuffer]=
simxCallScriptFunction(obj,clientID,scriptDescription,options,functionName,inInts,inFloats,inStri
ngs,inBuffer,operationMode)

    if (length(inStrings)>0)
        if (inStrings(length(inStrings))~=0)
            inStrings=[inStrings 0];
        end
    end
    strCnt=0;
    for i = 1:length(inStrings)
        if (inStrings(i)==0)
            strCnt=strCnt+1;
```

```
            end
        end
        variousIntsIn_ =
libpointer('int32Ptr',int32([clientID,options,numel(inInts),numel(inFloats),strCnt,numel(inBuffer),o
perationMode]));
        scriptDescriptionAndFunctionName_ = libpointer('int8Ptr',int8([scriptDescription 0
functionName 0]));
        inInts_ = libpointer('int32Ptr',int32(inInts));
        inFloats_ = libpointer('singlePtr',single(inFloats));
        inStrings_ = libpointer('int8Ptr',int8(inStrings));
        inBuffer_ = libpointer('uint8Ptr',uint8(inBuffer));


calllib(obj.libName,'mtlb_simxCallScriptFunction_a',variousIntsIn_,scriptDescriptionAndFunction
Name_,inInts_,inFloats_,inStrings_,inBuffer_);

        variousIntsOut = libpointer('int32Ptr',int32([0,0,0,0]));
        outInts_ = libpointer('int32PtrPtr');
        outFloats_ = libpointer('singlePtrPtr');
        outStrings_ = libpointer('int8PtrPtr');
        outBuffer_ = libpointer('uint8PtrPtr');

        [rtn  variousIntsOut outInts_ outFloats_ outStrings_ outBuffer_ ] =
calllib(obj.libName,'mtlb_simxCallScriptFunction_b',clientID,variousIntsOut,outInts_,outFloats_,o
utStrings_,outBuffer_);

        outIntCnt_ = variousIntsOut(1);
        outFloatCnt_ = variousIntsOut(2);
        outStringCnt_ = variousIntsOut(3);
        outBufferSize_ = variousIntsOut(4);

        if (rtn==0)
            if (outIntCnt_~=0)
                outInts_.setdatatype('int32Ptr',1,outIntCnt_);
                retInts = outInts_.value;
            else
                retInts=[];
            end

            if (outFloatCnt_~=0)
                outFloats_.setdatatype('singlePtr',1,outFloatCnt_);
                retFloats = outFloats_.value;
            else
```

```
            retFloats=[];
        end

        if (outStringCnt_>0)
            s=1;
            outStrings_.setdatatype('int8Ptr',1,s);
            charValue = outStrings_.value(s);
            while (outStringCnt_>0)
                while(charValue ~= 0)
                    s=s+1;
                    outStrings_.setdatatype('int8Ptr',1,s);
                    charValue = outStrings_.value(s);
                end
                outStringCnt_=outStringCnt_-1;
                if (outStringCnt_>0)
                    s=s+1;
                    outStrings_.setdatatype('int8Ptr',1,s);
                    charValue = outStrings_.value(s);
                end
            end
            tmp = outStrings_.value(1:s);
            retStrings = char(tmp);
        else
            retStrings = [];
        end

        if (outBufferSize_~=0)
            outBuffer_.setdatatype('uint8Ptr',1,outBufferSize_);
            retBuffer = outBuffer_.value;
        else
            retBuffer=[];
        end
    else
        retInts=[];
        retFloats=[];
        retStrings=[];
        retBuffer=[];
    end
end


    function [rtn signalValue ]=
simxReadStringStream(obj,clientID,signalName,operationMode)
```

```
        signalName_ = libpointer('int8Ptr',[int8(signalName) 0]);
        signalValue_ = libpointer('uint8PtrPtr');
        signalLength= libpointer('int32Ptr',int32(0));
        operationMode_ = int32(operationMode);

        [rtn signalName_ signalValue_ signalLength ] =
calllib(obj.libName,'simxReadStringStream',clientID,signalName_,signalValue_,signalLength
,operationMode_);

        if (rtn==0)
            if (signalLength>0)
                signalValue_.setdatatype('uint8Ptr',1,double(signalLength));
                signalValue = char(signalValue_.value);
            else
                signalValue = [];
            end
        else
            signalValue = [];
        end
    end

    function [rtn ]=
simxWriteStringStream(obj,clientID,signalName,signalValue,operationMode)
        signalLength_ = int32(length(signalValue));
        signalName_ = libpointer('int8Ptr',[int8(signalName) 0]);
        signalValue_ = libpointer('uint8Ptr',[uint8(signalValue) 0]);
        operationMode_ = int32(operationMode);

        [rtn signalName_ signalValue_ ] =
calllib(obj.libName,'simxWriteStringStream',clientID,signalName_,signalValue_,signalLength_
,operationMode_);
    end

  end
end
```