```
//------------------------------------------------------------------------------------------------
/*! ¥file
    ¥brief  4-legged robot simulator - server
    ¥author Akihiko Yamaguchi
    ¥date   Mar.13 2007  */
//------------------------------------------------------------------------------------------------
#ifndef ODE_MINOR_VERSION
  #error ODE_MINOR_VERSION should be set in compile
  #error   ex. -DODE_MINOR_VERSION=10
#endif
//------------------------------------------------------------------------------------------------
#include <ode/ode.h>
#include <drawstuff/drawstuff.h>
#include <iostream>
#undef PACKAGE_BUGREPORT
#undef PACKAGE_NAME
#undef PACKAGE_STRING
#undef PACKAGE_TARNAME
#undef PACKAGE_VERSION
#include <octave/config.h>
#include <octave/Matrix.h>
//------------------------------------------------------------------------------------------------
#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
//------------------------------------------------------------------------------------------------
#include "protocol.h"
//------------------------------------------------------------------------------------------------
#ifdef _MSC_VER
#pragma warning(disable:4244 4305)  // for VC++, no precision loss complaints
#endif
// select correct drawing functions
#ifdef dDOUBLE
#define dsDrawBox dsDrawBoxD
#define dsDrawSphere dsDrawSphereD
#define dsDrawCylinder dsDrawCylinderD
#define dsDrawCapsule dsDrawCapsuleD
#define dsDrawConvex dsDrawConvexD
#endif
//------------------------------------------------------------------------------------------------
using namespace std;
//------------------------------------------------------------------------------------------------
#include "robot.cpp"
//------------------------------------------------------------------------------------------------


//================================================================================================
//! ¥brief ふたつのオブジェクト o1, o2 が衝突しそうならこのコールバック関数が呼ばれる
//! ¥note 衝突しているかいないかはこの関数で（ユーザが）判定し，衝突していれば接触点にリンクを追加する.
static void nearCallback (void *data, dGeomID o1, dGeomID o2)
//================================================================================================
{
  // exit without doing anything if the two bodies are connected by a joint
  dBodyID b1 = dGeomGetBody(o1);
  dBodyID b2 = dGeomGetBody(o2);
  if (b1 && b2 && dAreConnectedExcluding(b1,b2,dJointTypeContact)) return;

  dContact contact[MAX_CONTACTS];   // up to MAX_CONTACTS contacts per box-box
  for (int i=0; i<MAX_CONTACTS; i++)
  {
    contact[i].surface.mode = dContactBounce | dContactSoftCFM;
    contact[i].surface.mu = dInfinity;
    contact[i].surface.mu2 = 0;
    contact[i].surface.bounce = 0.1;
    contact[i].surface.bounce_vel = 0.1;
    contact[i].surface.soft_cfm = 0.01;
  }
  if (int numc = dCollide (o1,o2,MAX_CONTACTS,&contact[0].geom,sizeof(dContact)))
  {
    for (int i=0; i<numc; i++)
    {
      dJointID c = dJointCreateContact (world.id(),contactgroup.id(),contact+i);
      dJointAttach (c,b1,b2);
    }
  }
}
//------------------------------------------------------------------------------------------------


//================================================================================================
//! ¥brief start simulation - set viewpoint
static void start()
//================================================================================================
{
  #if ODE_MINOR_VERSION>=10
    dAllocateODEDataForThread(dAllocateMaskAll);
  #endif

  static float xyz[3] = {0.75,1.3,1.0};
  static float hpr[3] = {-120.0,-16.0,0.0};

  dsSetViewpoint (xyz,hpr);
  // cerr << "Press 'R' to reset simulation¥n" << endl;
}
//------------------------------------------------------------------------------------------------

//================================================================================================
//! ¥brief キーイベントのコールバック関数
//! ¥param[in] cmd  入力キー
static void keyEvent (int cmd)
//================================================================================================
```

```
106 {
107   // if (cmd=='r'||cmd=='R')
108   // {
109   //   create_world();
110   // }
111 }
112 //-------------------------------------------------------------------------------------------------
113
114 static void getJointState (double state[JOINT_STATE_DIM])
115 {
116   for (int j(0);j<JOINT_NUM;++j)
117   {
118     state[j] = joint[j].getAngle();
119     state[JOINT_NUM+j] = joint[j].getAngleRate();
120   }
121   // cerr<<"joint1= ";for(int j(0);j<JOINT_STATE_DIM;++j)cerr<<" "<<state[j];cerr<<endl;
122 }
123 //-------------------------------------------------------------------------------------------------
124
125 static void getBaseState (double state[BASE_STATE_DIM])
126 {
127   state[0]  = body[0].getPosition()[0];  // x
128   state[1]  = body[0].getPosition()[1];  // y
129   state[2]  = body[0].getPosition()[2];  // z
130   state[3]  = body[0].getQuaternion()[0]; // quaternion (w)
131   state[4]  = body[0].getQuaternion()[1]; // quaternion (x)
132   state[5]  = body[0].getQuaternion()[2]; // quaternion (y)
133   state[6]  = body[0].getQuaternion()[3]; // quaternion (z)
134
135   state[7]  = body[0].getLinearVel()[0];  // vx
136   state[8]  = body[0].getLinearVel()[1];  // vy
137   state[9]  = body[0].getLinearVel()[2];  // vz
138   state[10] = body[0].getAngularVel()[0] ; // wx
139   state[11] = body[0].getAngularVel()[1] ; // wx
140   state[12] = body[0].getAngularVel()[2] ; // wx
141 }
142 //-------------------------------------------------------------------------------------------------
143
144
145
146 //-------------------------------------------------------------------------------------------------
147 static int    global_file_descriptor(-1);
148 static int    window_x(400), window_y(400);
149 // static const  dReal time_step (0.0005); // シミュレーションきざみ幅(0.5[ms])
150 static ColumnVector  input_torque (JOINT_NUM, 0.0);
151 //-------------------------------------------------------------------------------------------------
152
153
154
155 void stepSimulation (const dReal &time_step)
156 {
157   for (int j(0); j<JOINT_NUM; ++j)
158     joint[j].addTorque(input_torque(j));
159   // cerr<<"torque= "<<input_torque.transpose()<<endl;
160
161   // シミュレーション
162   space.collide (0,&nearCallback);
163   world.step (time_step);
164   // time += time_step;
165   // remove all contact joints
166   contactgroup.empty();
167 }
168 //-------------------------------------------------------------------------------------------------
169
170
171 bool oct_robot_server (void)
172 {
173   TXData data;
174   while (1)
175   {
176     if (global_file_descriptor<0)
177     {
178       cerr<<"connection terminated (unexpected error)."<<data.command<<endl;
179       exit(1);
180     }
181     read (global_file_descriptor, (char*)&data, sizeof(data));
182     switch (data.command)
183     {
184       case  ORS_START_SIM        :
185         return true;
186       case  ORS_STOP_SIM         :
187         return false;
188       case  ORS_STEP_SIM         :
189         stepSimulation (data.dvalue);
190         break;
191       case  ORS_RESET_SIM        :
192         create_world();
193         break;
194       case  ORS_DRAW_WORLD       :
195         return true;
196       case  ORS_SET_TORQUE       :
197         input_torque(data.step) = data.dvalue;
198         break;
199       case  ORS_SET_WINDOWSIZE   :
200         if (data.step==0)  window_x=data.ivalue;
201         else if (data.step==1)  window_y=data.ivalue;
202         break;
203       case  ORS_GET_JOINT_NUM    :
204         write (global_file_descriptor, (char*)&JOINT_NUM, sizeof(JOINT_NUM));
205         break;
206       case  ORS_GET_JSTATE_DIM   :
207         write (global_file_descriptor, (char*)&JOINT_STATE_DIM, sizeof(JOINT_STATE_DIM));
208         break;
209       case  ORS_GET_BSTATE_DIM   :
210         write (global_file_descriptor, (char*)&BASE_STATE_DIM, sizeof(BASE_STATE_DIM));
```

```
211         break;
212       case  ORS_GET_JOINT_STATE  :
213         getJointState (joint_state);
214         // cerr<<"joint2= ";for(int j(0);j<JOINT_STATE_DIM;++j)cerr<<" "<<joint_state[j];cerr<<endl;
215         write (global_file_descriptor, (char*)joint_state, sizeof(double)*JOINT_STATE_DIM);
216         break;
217       case  ORS_GET_BASE_STATE   :
218         getBaseState (base_state);
219         write (global_file_descriptor, (char*)base_state, sizeof(double)*BASE_STATE_DIM);
220         break;
221       default :
222         cerr<<"in oct_robot_server(): invalid command "<<data.command<<endl;
223         return false;
224     }
225   }
226 }
227 //-------------------------------------------------------------------------------------------
228
229 void setup_server (void)
230   // ref. http://www.ueda.info.waseda.ac.jp/~toyama/network/example1.html
231 {
232   int    fd1;
233   struct sockaddr_un    saddr;
234   struct sockaddr_un    caddr;
235   int    len;
236
237   if ((fd1 = socket (PF_UNIX, SOCK_STREAM, 0)) < 0)
238   {
239     perror("socket");
240     exit(1);
241   }
242
243   bzero ((char *)&saddr, sizeof(saddr));
244   // ソケットの名前を代入
245   saddr.sun_family = AF_UNIX;
246   strcpy (saddr.sun_path, SOCK_NAME);
247   // ソケットにアドレスをバインド
248   unlink(SOCK_NAME);
249   if (bind(fd1, (struct sockaddr *)&saddr,
250           sizeof(saddr.sun_family) + strlen(SOCK_NAME)) < 0){
251     perror("bind");
252     exit(1);
253   }
254   // listen をソケットに対して発行
255   if (listen(fd1, 1) < 0)
256   {
257     perror("listen");
258     exit(1);
259   }
260
261   len = sizeof(caddr);
262   /*
263     * accept()により、クライアントからの接続要求を受け付ける。
264     * 成功すると、クライアントと接続されたソケットのディスクリプタが
265     * fd2に返される。このfd2を通して通信が可能となる。
266     * fd1は必要なくなるので、close()で閉じる。
267     */
268   if ((global_file_descriptor = accept(fd1, (struct sockaddr *)&caddr, (socklen_t*)&len)) < 0)
269   {
270       perror("accept");
271       exit(1);
272   }
273   close(fd1);
274 }
275 //-------------------------------------------------------------------------------------------
276
277
278 //===========================================================================================
279 /*! ¥brief 描画(OpenGL)のコールバック関数.
280     ¥param[in] pause 停止モードなら true (0以外)
281
282     シミュレーションのきざみ time_step=0.0005[s] に対して描画は 50 fps 程度で十分なので,
283     1 frame ごとに simStepsPerFrame=1.0/time_step/FPS=40 回ダイナミクスのシミュレーションを回す.  */
284 static void simLoop (int pause)
285 //===========================================================================================
286 {
287 //   static dReal time(0.0); // シミュレーション時間
288   if (!pause)
289   {
290     if (!oct_robot_server())  dsStop();
291   }
292
293   draw_world();
294 }
295 //-------------------------------------------------------------------------------------------
296
297 static void stopSimulation (void)
298 {
299   close (global_file_descriptor);
300   global_file_descriptor = -1;
301 }
302 //-------------------------------------------------------------------------------------------
303
304
305 int main (int argc, char **argv)
306 {
307   dsFunctions fn; // OpenGL 出力用オブジェクト
308   fn.version = DS_VERSION;
309   fn.start = &start;
310   fn.step = &simLoop;
311   fn.command = &keyEvent;
312   fn.stop = &stopSimulation;
313   char path_to_textures[]="textures";
314   fn.path_to_textures = path_to_textures;  //! ¥note カレントディレクトリに textures へのリンクが必要
315
```

```
316   #if ODE_MINOR_VERSION>=9
317   # if ODE_MINOR_VERSION==9
318     dInitODE();
319   # elif ODE_MINOR_VERSION>=10
320     dInitODE2(0);
321   # endif
322   #endif
323
324   setup_server();
325   if (oct_robot_server())
326   {
327     create_world();
328     // run simulation
329     if(window_x>0 && window_y>0)
330       dsSimulationLoop (argc,argv,window_x,window_y,&fn);
331     else
332       while(oct_robot_server());
333   }
334   if(global_file_descriptor!=-1)
335   {
336     close (global_file_descriptor);
337     global_file_descriptor = -1;
338   }
339
340   #if ODE_MINOR_VERSION>=9
341     dCloseODE();
342   #endif
343   return 0;
344 }
345 //-----------------------------------------------------------------------------------------
346
```