```
//--------------------------------------------------------------------------------
/*! ¥file
    ¥brief  create 4-legged robot for ODE
    ¥author Akihiko Yamaguchi
    ¥date   Mar.20 2007 */
//--------------------------------------------------------------------------------
// dynamics and collision objects
static dWorld world;
static dSimpleSpace space (0);
static dPlane plane;
static dBody body[9];
static const int JOINT_NUM(8);
static const int JOINT_STATE_DIM(JOINT_NUM*2);  // 関節状態の次元
static const int BASE_STATE_DIM(13);  // ベース状態の次元
static dHingeJoint joint[JOINT_NUM];
static dJointGroup contactgroup;
static dBox  LinkTorso;
static dCapsule LinkLeg[8];

static const int  MAX_CONTACTS (10);  // maximum number of contact points per body
//--------------------------------------------------------------------------------
static double joint_state[JOINT_STATE_DIM];
static double base_state[BASE_STATE_DIM];
//--------------------------------------------------------------------------------
static const dReal _scale       = 0.5;
static const dReal param_h0     = 0.1    * _scale;
static const dReal param_wx0    = 1.6    * _scale;
static const dReal param_wy0    = 0.8    * _scale;
static const dReal param_px     = 0.14   * _scale;
static const dReal param_py     = 0.10   * _scale;
static const dReal param_d1     = 0.15   * _scale;
static const dReal param_l1     = 0.5    * _scale;
static const dReal param_d2     = 0.15   * _scale;
static const dReal param_l2     = 0.5    * _scale;
static const dReal param_dj     = 0.25   * _scale;

static const dReal density      = 2000.0;  // 各リンクの密度[kg/m^3]. 参考(?)`人体の密度' は 900~1100 kg/m^3 (wikipedia)
//--------------------------------------------------------------------------------


//================================================================================
//! ¥brief シミュレーションオブジェクトを作成
void create_world (void)
//================================================================================
{
  int j;
  contactgroup.create (0);
  world.setGravity (0,0,-9.8);  // 重力 [m/s^2]
  dWorldSetCFM (world.id(),1e-5);
  plane.create (space,0,0,1,0); // 地面（平面）.

  const dReal cx=0.0, cy=0.0, cz=param_l1+param_l2;
  j=0; { // 胴体
    body[j].create (world);
    body[j].setPosition (cx, cy, cz);
    dReal xx=param_wx0, yy=param_wy0, zz=param_h0;
    dMass m;
    m.setBox (density,xx,yy,zz);
    body[j].setMass (&m);
    LinkTorso.create (space,xx,yy,zz);
    LinkTorso.setBody (body[j]);
  }
  for (int k(0); k<4; ++k)
  {
    dReal xx, yy, zz;
    if (k==0 || k==1)  xx=cx+0.5*param_wx0-param_px;
    else               xx=cx-0.5*param_wx0+param_px;
    if (k==0 || k==2)  yy=cy+0.5*param_wy0+param_py;
    else               yy=cy-0.5*param_wy0-param_py;
    // 脚
    for (int i(0); i<2; ++i)
    {
      j=2*k+i+1;
      dReal rad, len;
      if (i==0) {rad=0.5*param_d1; len=param_l1-2.0*rad; zz=cz-0.5*param_l1;}
      else      {rad=0.5*param_d2; len=param_l2-2.0*rad; zz=cz-param_l1-0.5*param_l2;}
      body[j].create (world);
      body[j].setPosition (xx, yy, zz); // リンク1の中心座標
      dMass m;
      m.setCapsule (density,3,rad,len);  // direction(3): z-axis
      body[j].setMass (&m);
      LinkLeg[j-1].create (space,rad,len);
      LinkLeg[j-1].setBody (body[j]);
    }
    // 関節
    dBodyID b1, b2;
    for (int i(0); i<2; ++i)
    {
      j=2*k+i;
      if (i==0)  {b1=body[0]; b2=body[j+1]; zz=cz;}
      else       {b1=body[j]; b2=body[j+1]; zz=cz-param_l1;}
      joint[j].create (world);
      joint[j].attach (b1,b2);
      joint[j].setAnchor (xx,yy,zz); // 回転中心=支柱の中心(=原点)
      joint[j].setAxis (0.0,1.0,0.0); // 回転軸=y軸
      joint[j].setParam (dParamHiStop, +0.5*M_PI); // 関節の可動範囲を制約するときに使う
      joint[j].setParam (dParamLoStop, -0.5*M_PI); // acrobot の場合は省略
    }
  }
}
//--------------------------------------------------------------------------------


//================================================================================
//! ¥brief 描画関数
```

```
106 void draw_world( void )
107 //=================================================================================================
108 {
109   int j;
110   dsSetColor (0,0.5,1);
111   dsSetTexture (DS_WOOD);
112   dReal rad, len;
113   dReal sides[4];
114   dVector3 pos;
115   dBox *blink;
116   dCapsule *clink;
117   dsSetTexture (DS_NONE);
118   dsSetColorAlpha (1.0, 1.0, 1.0, 0.8);
119   j=0; blink=&LinkTorso; blink->getLengths(sides); dsDrawBox (blink->getPosition(),blink->getRotation(),sides);
120   for (j=1; j<=8; ++j)
121     {clink=&LinkLeg[j-1]; clink->getParams(&rad, &len); dsDrawCapsule (clink->getPosition(),clink->getRotation(),len,rad);}
122   dsSetColorAlpha (0.0, 1.0, 0.0, 0.6);
123   for (j=0; j<8; ++j)
124     {joint[j].getAnchor(pos); dsDrawSphere (pos, body[0].getRotation(), 0.5*param_dj);}
125 }
126 //-------------------------------------------------------------------------------------------------
127
128
129
130
```