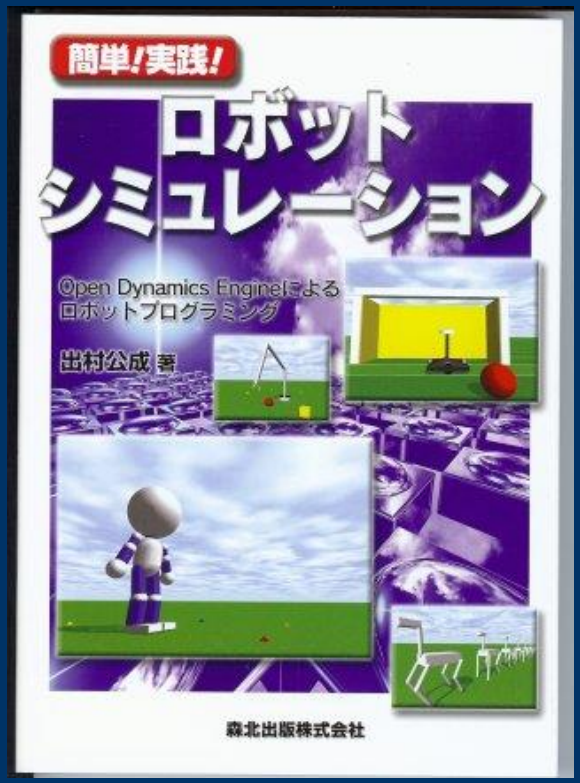
ロボットシミュレーション

Open Dynamics Engineによるロボットプログラミング

Part1: Open Dynamics Engine



2009-1-9　2版
2008-7-9　初版

出村　公成（でむらこうせい）
[Web] http://demura.net
[Mail] info@demura.net

非営利的な教育目的であれば，このパワーポイントを自由にお使いください。

簡単！実践！ロボットシミュレーションの表紙

このパワーポイントについて

- このパワーポイントは以下の書籍を講義等で教えるため出村公成によって作られたものです。
  - 出村公成著
  - 簡単！実践！ロボットシミュレーション Open Dynamics Engineによるロボットプログラミング
  - 森北出版、2007
- 非営利的な教育目的であれば自由に使ってください。ただし、著者の名前とこのスライドは削除しないようお願いします。

内　容

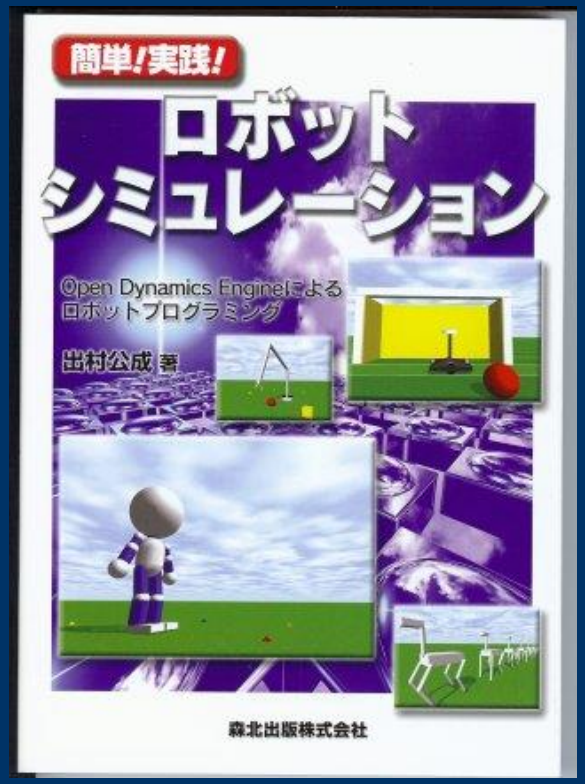
- 教科書
  - Step1: ODE初体験
    - 概　要
    - 開発環境･ODEのインストール
    - 動力学、衝突検出計算
    - 基本的なAPIの説明

１週目

- ODEの概要
- 開発環境、ODEのインストール
- 動力学計算
- 演習

教科書

- 簡単！実践！ロボットシミュレーション
  Open Dynamics Engineによるロボットプログラミング
- 出村公成著
- 森北出版
- 2007年5月
- ISBN-13: 978-4627846913



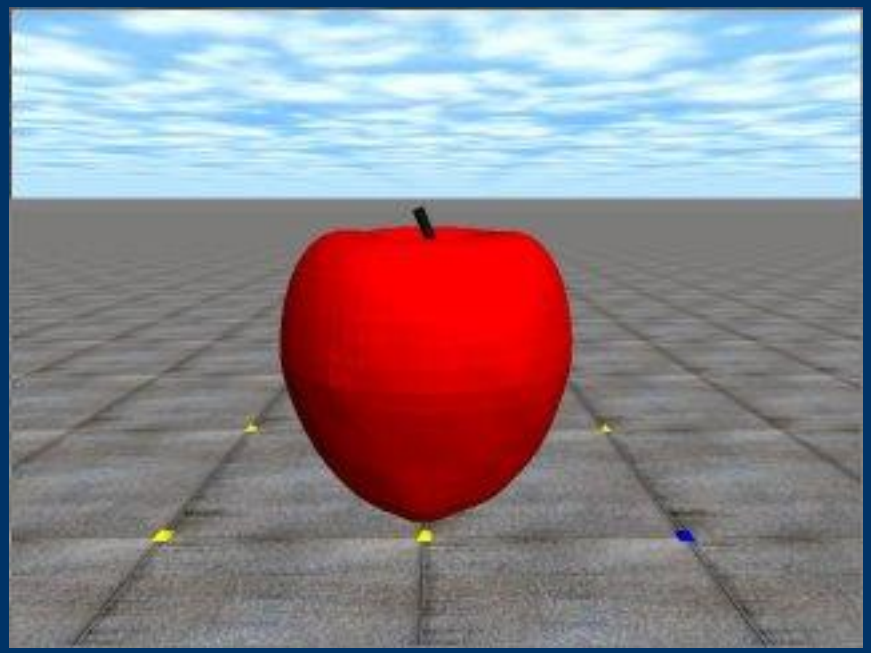
簡単！実践！ロボットシミュレーションの表紙

教科書の特徴

- ロボット工学の基礎とそのプログラミングを学べる実践書
- 日本初のODE解説書
  - 総合解説書としては世界初
- ODEプロジェクトへの寄付
  - 教科書売り上げ１冊につき1ドルをODEプロジェクトへ寄付
- お財布にやさしく
  - 教科書以外はお金がかからない
- WEBサイトとの連動
  - http://demura.netからソースコードのダウンロード可能

目　次

Step1 ODE初体験



ODEの概要

- ラッセル・スミス博士らが2001年から開発を続けている剛体動力学計算エンジン
- Open Dynamics Engine
- 公式Webサイト
  - http://www.ode.org
- 特　徴
  - オープンソース
  - 簡単なので使いやすい
  - 高速で安定性がある
- 用　途
  - ゲームの物理計算エンジン
  - バーチャルリアリティ，ロボットのシミュレータ
    - 精度が要求される用途には向かない

ODEの技術的な特徴

- 任意の質量分布を持つ剛体のシミュレーションが可能
- ジョイント：ボール，ヒンジ，スライダー，ユニバーサル
- 衝突プリミティブ：球，直方体，カプセル，円柱，平面，ポリゴン
- 動力学計算：ニュートン・オイラー法
  - 高速で安定性はあるが精度は高くない
- 衝突検出機能が組み込まれている
- C, C++でプログラミング可能
- Windows, Linux, Mac OS Xで使用可能

ODEのライセンス

- ソフトウェアを使用する場合はライセンスを確認することが重要。ライセンス違反は個人並びに組織に大きな損害をもたらす。
- ２重ライセンス
  - どちらか好きな方のライセンスを利用可能
  - これらにより，商用利用してもソースコードを公開する必要なし
- GNU Lesser Public License (LGPL)
  - ライブラリ用のGPL
  - ライセンスの全文
    - http://www.gnu.org/licenses/lgpl.html
    - http://www.gnu.org/licenses/lgpl.ja.html (日本語訳)
- BSD Style License
  - 著作権表示、無保証、免責だけが条件の寛大なライセンス
  - ライセンスの全文
    - http://opende.sourceforge.net/ode-license.html

GPL (GNU General Public License)

ソフトウェアの自由な実行、複製、改変および再配布を、「無保証」の条件付きで許諾する利用許諾（ライセンス）、またはその契約文書のこと。FSFによって考案された。

ソフトウェアなどの著作者が、自身の著作権の保持を表明したまま、複製や改変などの多くの著作支分権の実施を複製物の受領者に許諾するとともに、その受領者が複製物またはそれらを改変した派生物を再配布することを妨げないこと、ソフトウェアの使用に伴う損害を保証しないことなどを条件に、ソフトウェアの利用を許諾する契約。また、ソフトウェアのバイナリ形式のみでの配布を認めず、ソースコードの添付またはソースコードの入手方法の明示を義務づけている。

1991年に発表されたVer.2は、長期にわたり多くのソフトウェアに採用されてきた。GPLのこの特定の版を指してGPLv2と表記されることがある。

2007年6月、FSFはGPL Ver.3（GPLv3）を発表した。この新しい版には、ソフトウェアの自由を守るという理念はそのままに、契約文書としての体裁を整えるとともに、近年のソフトウェアの利用形態に合わせた条件の見直しなどが含まれている。

オープンソース情報データベース　OSS iPediaから転載
http://ossipedia.ipa.go.jp/kb/GNU_General_Public_License

GNU Lesser Public License (LGPL)

「ほかのソフトウェアと組み合わされて実行されたり配布されたりするソフトウェアを対象に、GPLに準ずる形式でソフトウェアなどの著作物の利用を許諾する契約、またはその契約文書のこと。FSFによって考案された。

GPLでは、そのライセンスの下で配布されるソフトウェアと組み合わされて配布される派生ソフトウェアについてもGPLが適用されるため、無制限な再配布 の保証やソースコードの開示といった「厳しい」制約が派生物にも課せられる。この制約を避けるため、LGPLではそのライセンスの下にあるソフトウェアと 一緒に配布されるソフトウェアについては、LGPLの制約は受けないという条項が設けられている。

もともとは、コンパイラなどでリンクして利用するライブラリなど、ほかのプログラムと組み合わされて利用されるソフトウェアを対 象にしたライセンスで、「商用ソフトウェア」でのフリーソフトウェアの利用を妨げないようにすることを目 的に考案されたライセンスで、当初は“GNU Library General Public License”と呼ばれていた。」

オープンソース情報データベース　OSS iPediaから転載

http://ossipedia.ipa.go.jp/kb/GNU_Lesser_General_Public_License

BSD License

カリフォルニア大学バークレー校でUNIX互換ソフトウェアなどの配布に使用されたライセンスのこと。

1999年7月22日の改正により、原作者に対する謝辞の明記を義務づけた「謝辞条項」（「宣伝条項」とも呼ばれる）が廃止され、現在のライセンス条文になった。改正後のものを、特に「New BSD License」（修正BSDライセンス）と呼ぶこともある。

オープンソース情報データベース　OSS iPediaから転載
http://ossipedia.ipa.go.jp/kb/BSD_License

New BSD License

カリフォルニア大学バークレー校のコンピュータサイエンス学科で開発されたBSD UNIXのオープンソース版である「4.3 BSD Net/2」から派生したBSD系UNIXシステム（FreeBSD、NetBSDなど）の配布で使われているライセンス。「バークレイスタイルライセンス（berkeley-style license）」とも。**著作権の表示と、無保証、免責のみを条項とする、制限の少ないライセンス方式である。**

元々の「BSDライセンス」には、開発者に対する謝辞を表示することを派生物にも強制する「謝辞条項」（「宣伝条項」とも呼ばれる）があったため、GPLを適用するソフトウェアと組み合わせた派生物の配布に制限が生ずるという問題があった。この条項を削除した現在のBSDライセンスのことを、特に「New BSD License」（修正BSDライセンス）と呼んでいる。

オープンソース情報データベース　OSS iPediaから転載

http://ossipedia.ipa.go.jp/kb/New_BSD_License

BSD style license for ODE

Open Dynamics Engine
Copyright (c) 2001-2004, Russell L. Smith.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the names of ODE's copyright owner nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

ODEソースコードのLICENSE-BSD.TXTの全文を転載

開発環境、ODEのインストール

- http://demura.net/robotsimu/
  の設定に従い開発環境、ODEをインストールしよう。
- 講義ではWindows上でCode::Blocksを使用
  - Code::BlocksはWindows, Linuxなどで利用可能なC/C++言語の統合開発環境
  - オープンソース
- 大まかな手順
  - 開発環境のインストール
    - Code::Blocks
  - ODEのインストール

動力学計算の手続き

1. ODEの初期化　dInitODE()
2. ワールドの生成　dWorldCreate()
3. 重力の設定　dWorldSetGravity()
4. 剛体（ボディ）の生成　dBodyCreate()
5. 質量パラメータの設定
   - dMassSetZero()
   - dMassSet***Total()
   - dBodySetMass()
6. 位置の設定
   - dBodySetPosition()
7. シミュレーションループ
   - 動力学計算　dWorldStep()
8. ワールドの破壊　dWorldDestroy()
9. ODEの終了　dCloseODE()

簡単のために引数等は省略

ODEの用語

- **ワールド (world)**
  - 動力学計算を受けるものを入れるソフトウェアの入れ物
- **ボディ (body)**
  - 剛体(rigid body)のこと.
    - 変形しない理想的な物体
  - ODEは剛体の動力学計算エンジン

1.3 リンゴの落下　P5

- **P7のプログラム1.1 hello.cppをmain関数から説明する**
- **プログラムはmain関数から実行**

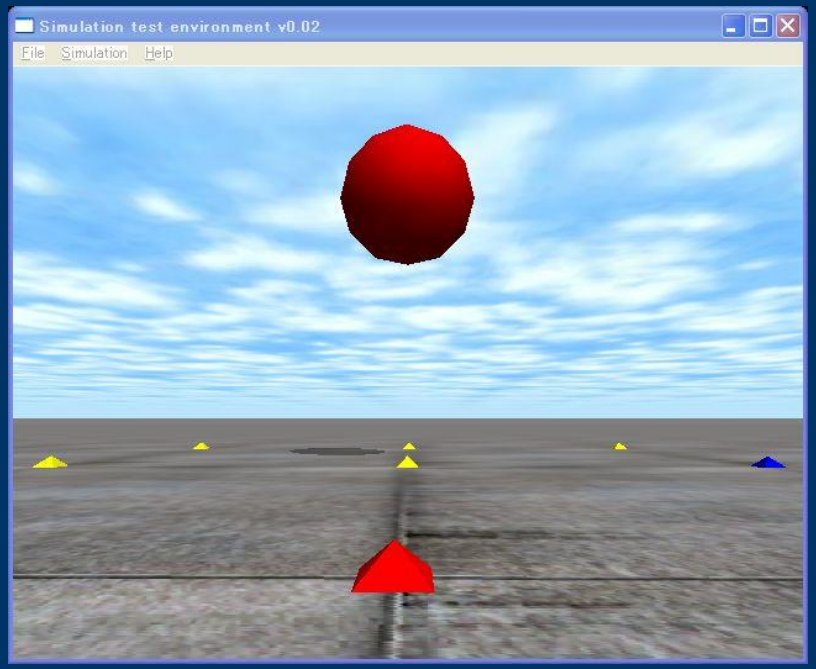


図1.1 リンゴの落下

プログラム1.1の変更

- **重要:ODE0.10からのAPI変更に伴い以下の追加が必須です。**
  - **dInitODE()を37行目の上に追加**
  - **dCloseODE()を49行目の上に追加**

なお、サポートサイトからダウンロード可能なファイルは変更済みです。

ODEの座標系

- 直交座標系（右手系）
  - 数学、物理で良く使われる右手系
  - 原点：9個あるピラミッドの中央
  - X軸：原点から赤ピラミッド
  - Y軸：原点から青ピラミッド
  - Z軸：原点から上空
- 単位
  - SI単位系 (P59)
    - 長さ　m
    - 質量　kg
    - 時間　s
    - 力　　N

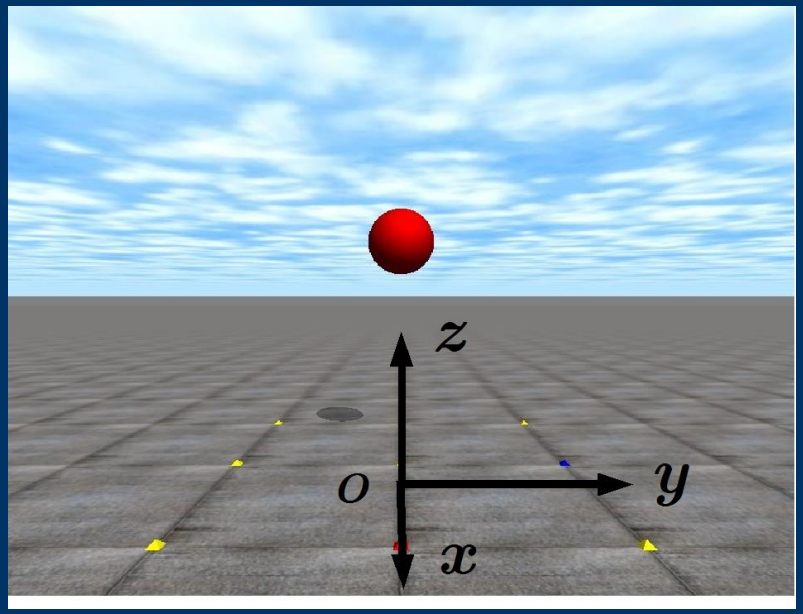


図1.2 ODEの座標系（教科書P9から転載）

カメラの設定

- 物体を画面に表示するためにカメラの視点と視線を設定
- 視点
  - static float xyz[3]={3.0, 0.0, 1.0};
  - 絶対座標系(3.0, 0.0, 1.0)
- 視線
  - static float hpr[3]={-180, 0, 0};
    - X軸の正方向からの視線
    - 単位は度[°], 他のAPIはラジアンなので注意
  - hprはheading, pitch, rollの頭文字
  - heading: z軸を中心に回転
  - pitch: y軸を中心に回転
  - roll: x軸を中心に回転

API: Application Program Interface

- ライブラリ付属の関数
- ODEのAPI
  - 小文字の**d**で始まる
- drawstuffのAPI
  - 3次元CG表示用のライブラリ，ODEにはもれなくオマケで付属
  - 小文字の**ds**で始まる

API：視点と視線の設定

- **void dsSetViewPoint(float xyz[3], float hpr[3]);**
  - **視点xyz[3], 視線hpr[3]にカメラを設定する**

API: 初期化と終了

- **ODEを使用する場合は以下のAPIで初期化し，終了しなければならない**
- **初期化**
  - **dInitODE()**
- **終了**
  - **dCloseODE()**

API：ワールドの生成

- **dWorldID dWorldCrate();**
  - 動力学計算の対象となるソフトウェアの入れ物ワールドを生成し，そのID番号を返す

API：重力の設定

- **void dWorldSetGravity(dWorldID world, dReal x, dReal y, dReal z);**
  - ワールドworldに重力加速度ベクトル(x, y, z)を設定する
  - 座標系は絶対座標系。ODEでは特に指示がない限り絶対座標系を使用。

API: 剛体の生成

- **dBodyID dBodyCreate(dWorldID world);**
  - **ワールドに剛体bodyを生成し，そのID番号を返す**

API: 質量パラメータの設定

- **void dMassSetZero(dMass *mass);**
  - 質量パラメータmassを0に初期化する
  - dMassは質量，重心位置，慣性モーメントから構成される構造体
- **void dMassSetSphereTotal(dMass *mass, dReal m, dReal r);**
  - 質量m, 半径rの球の質量パラメータを計算し，massに設定する
- **void dBodySetMass(dBodyID body, const dMass *mass);**
  - 質量パラメータmassをボディbodyに設定する

API: 位置の設定

- **void dBodySetPosition(dBodyID body, dReal x, dReal y, dReal z);**
  - **ボディbodyの位置を絶対座標系で(x, y, z)に設定する**

API: 動力学計算

- **void dWorldStep(dWorldID world, dReal stepsize);**
  - ワールドworldの動力学計算をstepsize[s]だけ進める
  - stepsizeは数値積分（教科書P127参照）の刻み幅
    - stepsizeを大きくすると速度は速くなるが計算精度は低下
    - stepsizeを小さくすると速度は遅くなるが計算精度は向上

精度と速度はトレードオフの関係

API：位置と姿勢の取得

- **const dReal *dBodyGetPosition(dBodyID body);**
- **const dReal *dBodyGetRotation(dBodyID body);**
  - ボディbodyの位置及び姿勢を取得する．前者は位置が格納されている配列へのポインタ，後者は姿勢（回転行列）が格納されている配列へのポインタを返す．

API: ワールドの破壊

- **void dWorldDestroy(dWorldID world);**
  - **ワールドworldを破壊する.**

エクササイズ　P12, 13

以下の問題をやろう

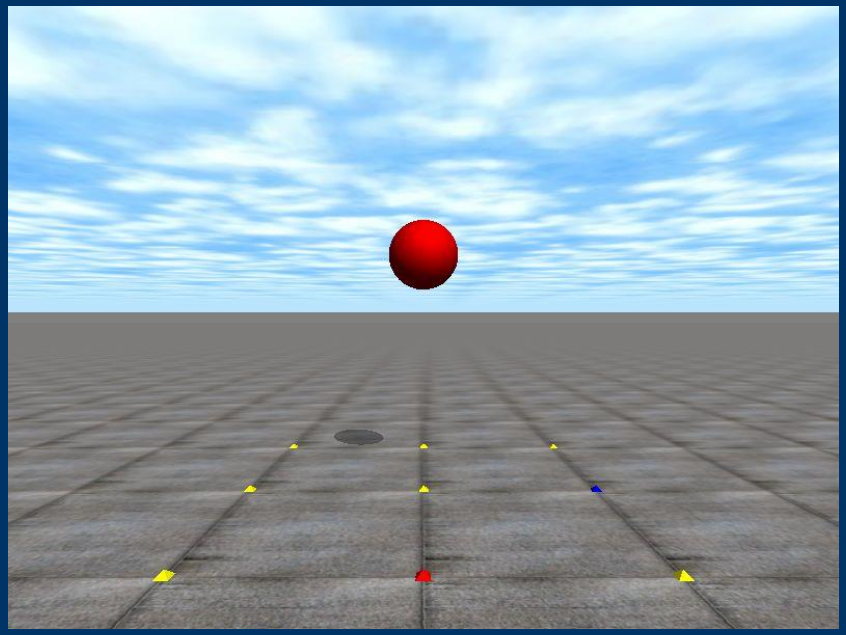
- EX1.1
  - 教科書の説明に従って開発環境とODEをインストールしよう
- EX1.2
- EX1.3
- EX1.4

2週目

- 衝突検出計算
- 演習

1.4 ボールの跳ね返り　P13

- ソースコードを説明する



ボールの跳ね返り

衝突検出の手続き

- 衝突検出用スペースの生成
  - dHashSpaceCreate();
- 接触点グループの生成
  - dJointGroupCreate();
- 衝突検出用ジオメトリの生成
  - dCreate***();
- ボディとジオメトリの関連付け
  - dGeomSetBody();
- シミュレーションループ
  - 衝突検出関数の呼び出し　dSpaceCollide();
  - 接触点グループを空に　　dJointGroupEmpty();
- スペースの破壊
  - dSpaceDestroy();

ODEの用語

- スペース(space)
  - 衝突検出計算の対象を入れるソフトウェアの入れ物
- ジオメトリ (geomtry)
  - 衝突検出計算を受ける物体の属性.
    ジオメトリとは形状の意味.

物体の属性 P14

- ボディ
  - 動力学計算の対象
- ジオメトリ
  - 衝突検出計算の対象

図1.4 物体の2つの属性（教科書P14から転載）

API: スペースの生成 P17

- **dSpaceID dHashSpaceCreate(dSpaceID space);**
  - 高速に計算可能なハッシュテーブルを備えた衝突検出計算用スペースを生成し，そのID番号を返す．

API：ジョイントグループの生成

- **dJointGroupID dJointGroupCreate(0);**
  - ジョイントグループを生成する．引数は現在使われていない，バージョン間の互換性を保つために0を入れる．

ジオメトリの種類

- ジオメトリの基本形状として，球，直方体，円柱，カプセル，光線，ポリゴン，平面などが用意されている．

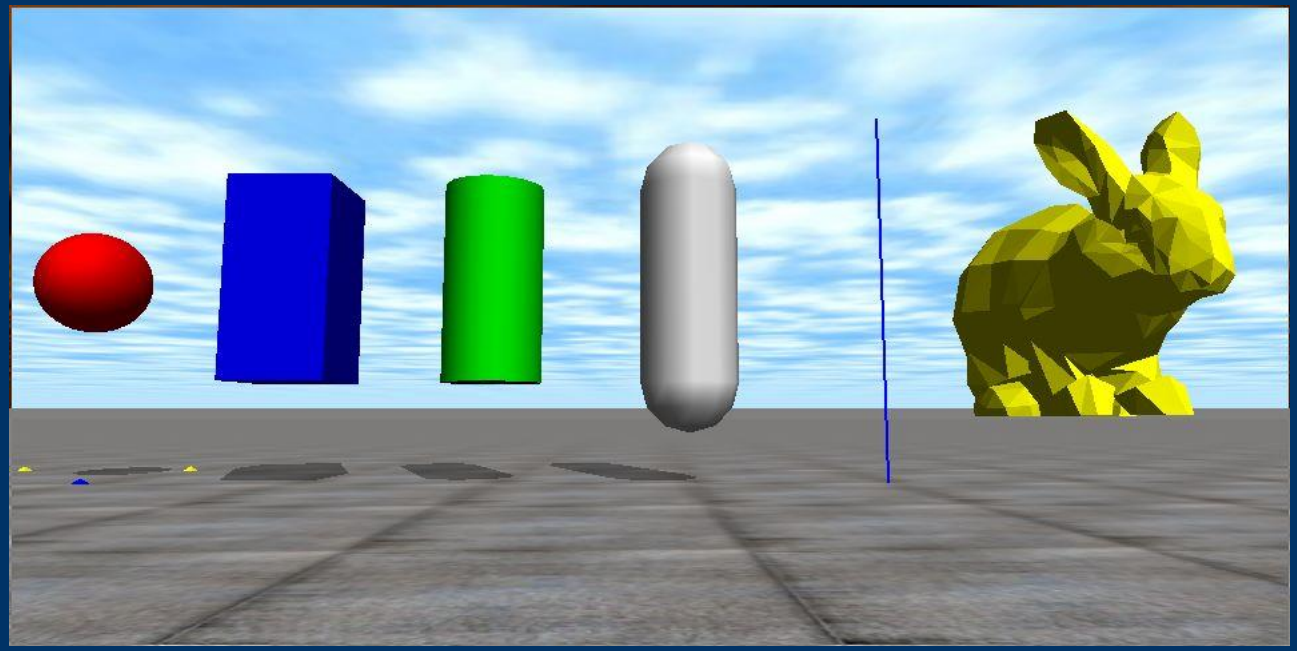


図3.3 ジオメトリの種類（教科書P64から転載）

API: 平面ジオメトリの生成　P18

- **dGeomID dCreatePlane(dSpaceID space, dReal a, dReal b, dReal c, dReal d);**
  - スペースに平面ジオメトリを生成する. 引数は平面の方程式ax + by + cz = dの各係数.

API: 球ジオメトリの生成　P18

- **dGeomID dCreateSphere(dSpaceID space, dReal r);**
  - スペースspaceに半径rの球ジオメトリを生成し，そのID番号を返す．

API：ジオメトリとボディの関連付け

- **void dGeomSetBody(dGeomID geom, dBodyID body);**
  - ジオメトリgeomとボディbodyを関連付ける．これにより，両者の位置ベクトルと回転行列は自動的に結び付けられる．

API: 衝突検出関数

- **void dSpaceCollide(dSpaceID space, void *data, dNearCallback *callback);**
  - 衝突しそうな2つのジオメトリペアを探し，コールバック関数callbackを呼び出す．引数dataはcallbackに渡すデータへのポインタ，データがないときは0を入れる．

API: ジョイントグループを空

- **void dJointGroupEmpty(dJointGroupID contactgroup);**
  - ジョイントグループを空にする. これには接触点達が格納されている.

API: 接触ジョイントの生成　P20

- **dJointID dJointCreateContact(dWorldID world, dJointGroupID contactgroup, const dContact *contact);**
  - 接触ジョイントを生成し，そのID番号を返す．引数contactgroupはすでに生成済みのものを割り当て，引数contactには接触点の属性が格納されるdContact構造体へのポインタを入れる．

API: ジョイントの取り付け

- **void dJointAttach(dJointID joint, dBodyID body1, dBodyID body2);**
  - ジョイント(関節)をボディ1とボディ2に取り付ける．ODEのジョイントは2つのボディ間に取り付けます．

エクササイズ P21

- **EX1.5**
- **EX1.6**
- **EX1.7**
- **EX1.8**
- **EX1.9**

3週目

- ジョイントの生成と簡単な制御
- キーボード操作
- 演習

1.5　1本脚ロボットを作ろう　P22

- ソースコードを説明する

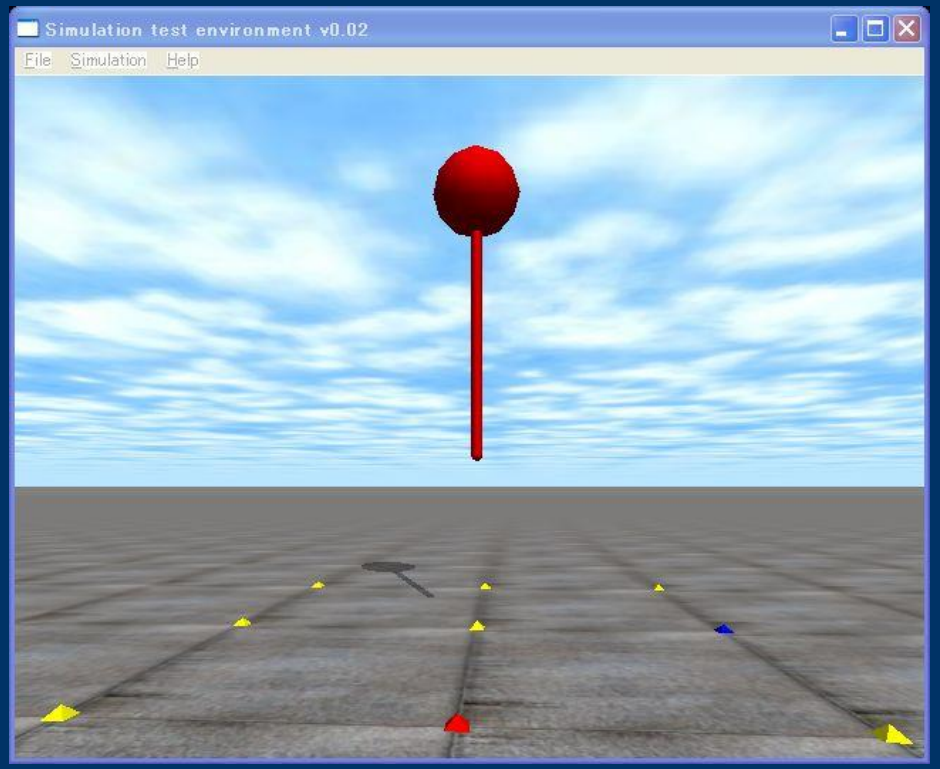


図1.6 monoBotの勇姿（教科書P23から転載）

ジョイントの使い方

- ジョイントの生成　dJointCreate***()
- ボディとの結合　dJointAttach()
- ジョイント中心点の設定
  - dJointSet***Anchor()
- ジョイント回転軸の設定
  - dJointSet***Axis()
- パラメータの設定
  - dJointSet***Param()

***にはジョイントの種類Ball, Hinge, Sliderなどが入る

ジョイントの種類　P84

- ボール
- ヒンジ
- スライダー
- ユニバーサル
- ヒンジ2
- 接触ジョイント

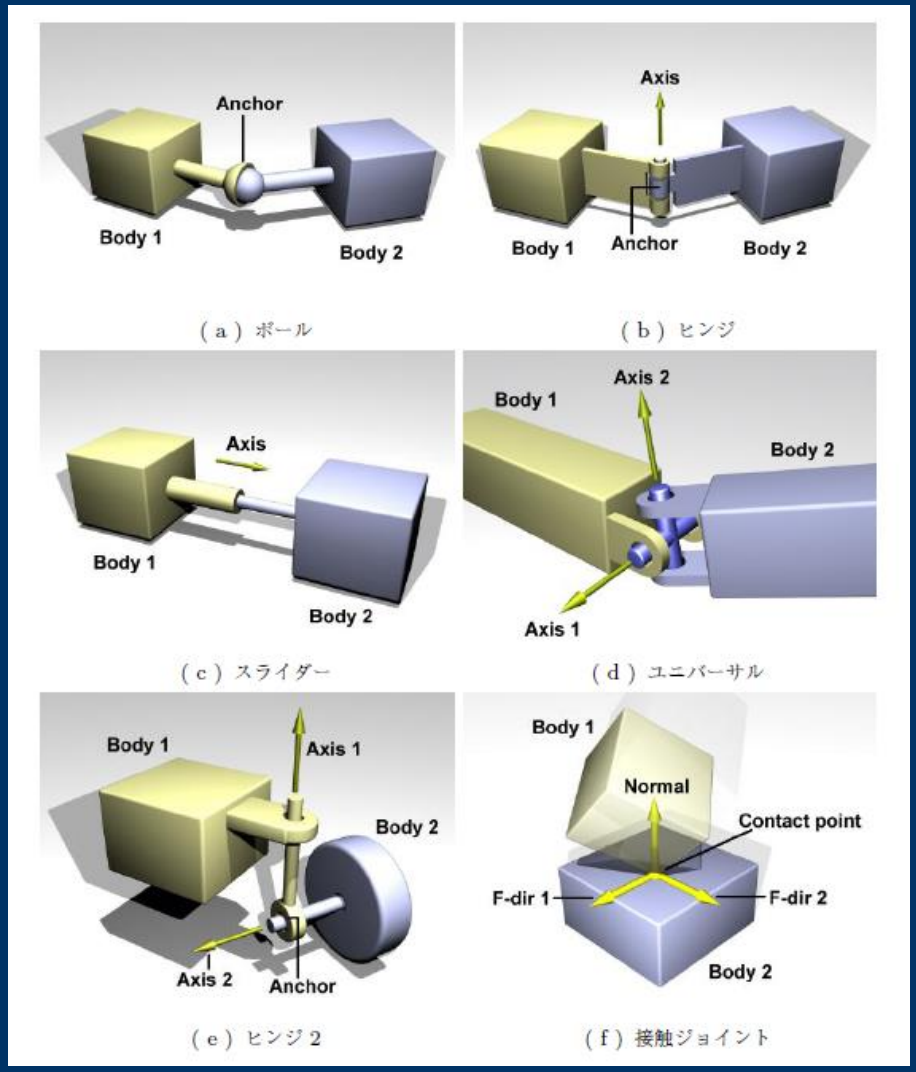


図3.9 ジョイントの種類（教科書P85から転載）

API:ヒンジジョイントの生成と破壊

- **dJointID dJointCreateHinge(dWorldID, dJointGroupID)**
  - ヒンジジョイントをworldに生成し，ID番号を返します．dJointGroupIDには0を通常設定します．
- **void dJointDestroy(dJointID joint);**
  - ジョイントを破壊します．

API: ヒンジジョイントの設定

- **void dJointSetHingeAnchor(dJointID, dReal x, dReal y, dReal z)**
  - ヒンジジョイントの中心点を絶対座標(x, y, z)に設定する
- **void dJointSetHingeAxis(dJointID, dReal x, dReal y, dReal z)**
  - ヒンジの回転軸ベクトル(x, y, z)を設定する
- **void dJointSetHingeParam(dJointID, int parameter, dReal value)**
  - ヒンジのパラメータparameterに値valueを設定する

ジョイントパラメータ　P97

dParamLoStop	**可動域の下限値** **デフォルト値　–dInfinity（-無限）** **-πより大きくないと無効**
dParamHiStop	**可動域の上限値** **デフォルト値　dInfinity（無限）** **πより小さくないと無効**
dParamVel	**モータの目標角速度（ヒンジ）または速度（スライダ）**
dParamFMax	**dParamVelを達成するために与える最大トルク（ヒンジ）または力（スライダ）**

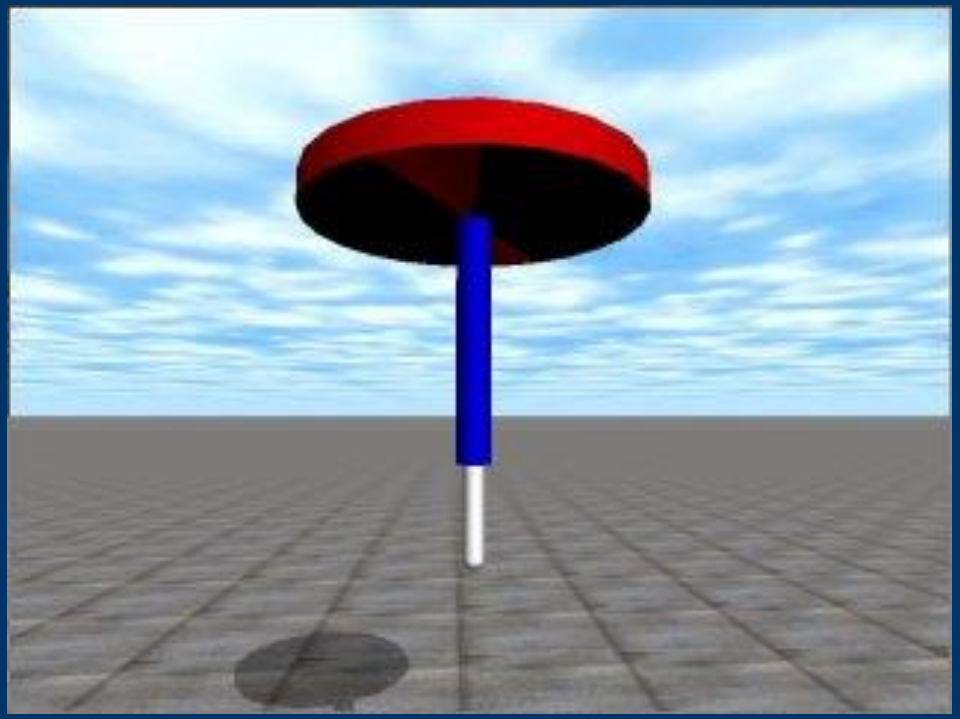
エクササイズ　P27

- **EX1.10**
- **EX1.11**
  をやりましょう！

プチプロジェクト　P27

- 必要に応じプチプロジェクトをやろう！

Step2 シミュレータを作ろう



2.1 ヒンジジョイントを動かすには

- ODEでは固定ジョイント以外はモータが内臓されています
- モータを動かす方法
  - 角速度を設定
  - それを達成するためのトルクを設定
- 角速度だけを設定してもトルクが0ならモータは動かない

プログラム2.1 ヒンジジョイントの制御

- P32のソースコードを説明する

API: ヒンジジョイント関連

- **dReal dJointGetHingeAngle(dJointID);**
  - ヒンジジョイントの角度[rad]を取得
- **void dJointSetHingeParam(dJointID, int parameter, dReal value);**
  - ヒンジジョイントのパラメータparameterをvalueに設定
  - parameter
    - dParamVel: 角速度[rad/s]
    - dParamFMax: トルク[N/m]

2.2 スライダージョイント

- 基本的にはヒンジジョイントと同じだが、スライダーは直線運動
- モータを動かす方法
  - 速度　[m/s]
  - 力　[N]

API スライダージョイント関連

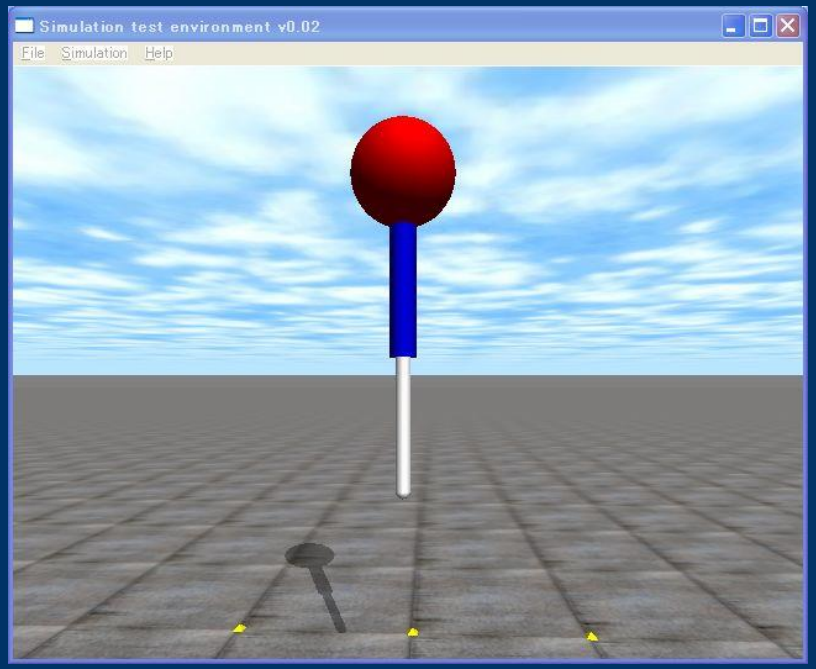
- **dJointID dJointCreateSlider(dWorldID, dJointGroupID);**
  - スライダジョイントを生成する.
- **void dJointSetSliderAxis(dJointID, dReal x, dReal y, dReal z);**
  - スライダジョイントの軸ベクトル(x, y, z)を設定する.
- **void dJointSetSliderParam(dJointID, int parameter, dReal value);**
  - スライダジョイントのパラメータを設定する.

エクササイズ　P33

- **EX2.1**
  をやりましょう！

2.3 ホッピングロボットを作ろう

- **P37のプログラム2.5を説明する。**



ODE特有のパラメータ　ERP

- **ERP: error reduction parameter**
  - 関節誤差修正パラメータ
    - 0以上1以下
    - デフォルト値　0.2
  - 計算誤差や外力などの影響による関節中心点のずれを修正。0の場合は修正なし、1の場合は次のステップで修正

ODE特有のパラメータ　CFM

- **CFM: constraint force mixing**
  - 拘束力混合パラメータ
    - 0以上1以下
    - デフォルト値
      - 単精度インストール　1E-5
      - 倍精度インストール　1E-10
  - 関節や地面などの柔らかさを調整。0の場合は剛体、大きくするとより柔らかくなる。

2.4 インタラクティブにいこう！

- キーボードを操作する方法
  1. コマンド関数名の設定
     - fn構造体のメンバcommandにコマンド関数のアドレスを代入
     - fn.command = &command;
  2. コマンド関数の作成
     - P41 プログラム2.7を説明する

2.4.3 シミュレーションの再実行　P43

1. 必要なオブジェクトの破壊
2. 接触点グループの破壊
3. 接触点グループの生成
4. 必要なオブジェクトの生成
   - ボディの生成
   - 質量、位置、姿勢の設定
   - ジオメトリの生成
   - ボディとジオメトリの対応付け
   - ジョイントの生成と設定

プログラム2.9

- **P44のプログラム2.9を説明する**

API: 力、トルクを加える

- **void dBodyAddForce(dBodyID body, dReal fx, dReal fy, dReal fz);**
  - ボディbodyに力ベクトル(fx, fy, fz)を加える
- **void dBodyAddTorque(dBodyID body, dReal fx, dReal fy, dReal fz);**
  - ボディbodyにトルクベクトル(fx,fy,fz)を加える。 fxはx軸まわりのトルク

エクササイズ

- **EX2.4**
- **EX2.5**
- **EX2.6**
- **EX2.7**
- **EX2.8**
  をやろう！

プチプロジェクト

- 簡単な物理ゲームを作ってみよう
  - ビリヤード
  - テトリスの3次元版
  - 3次元ブロック崩し
  - ドミノ倒し
  - …

おしまい。
Enjoy ODE !