Choreonoid講習会
～ロボットモデルの作成方法～

中村啓太（会津大学）
穴澤剛士（株式会社FSK）

講習会目的

❖ Choreonoidの基本操作を学ぶ

❖ シンプルコントローラの作成方法を学ぶ

❖ 自律制御について学び、ライントレースを行う

講習会内容

- ❖ Choreonoidとは?
- ❖ Choreonoidの基本操作
- ❖ プロジェクトの作成
- ❖ ボディモデルとは?
- ❖ コントローラとは?
- ❖ TurtleBot2の自律走行
- ❖ カメラ画像を用いたTurtleBot2の自律走行

ボディモデルとは?

ボディモデル

❖ Choreonoidで読み込むことができるモデルファイル

❖ モデルの形状，色，関節構造，剛体パラメータ，
センサ情報などを記述

❖ 記述方式は『**YAML**』形式を採用しており，
ファイル拡張子は『**.body**』として作成

❖ 編集を行う場合は，テキストエディタを起動し，編集

テキストエディタの起動方法

- ターミナル上でgeditコマンドを実行
  - ファイルの新規作成

    ```
    $ gedit &
    ```

  - 既存ファイルの編集

    ```
    $ gedit [ファイルパス] &
    ```

- 今回，以下のファイルを起動

  ```
  $ gedit ~/choreonoid/share/model/
  TurtleBotTurtleBot2.body &
  ```

  - **&**：バックグラウンドでの実行
    - 1つのターミナル上で，
      複数のエディタを開きたい場合などに付与

テキストエディタの操作方法

- ファイル保存：編集内容を保存
  - 右上の『保存』ボタン
    もしくは，『Ctrl』キーを押しながら，『S』キーを押下
- ハイライト表示：bodyファイルの構文に色を付け表示
  - メインメニューの『表示』→『ハイライトモード』を選択し，
    『YAML』を選択
- 行番号表示：行番号を表示
  - メインメニューの『編集』→『設定』を選択し，
    『表示』タブ→『行番号を追加する』にチェックを入れ，
    『×』で閉じる

モデルの基本構造

- モデルは複数のパーツで構成
  - TurtleBot2 モデルの基本構造

ヘッダ

- ファイルの先頭に記述されるファイルについての基本情報
  - ボディモデルファイルでは，以下の項目を記述
    - format：形式を指定
    - formatVersion：バージョン番号を指定
    - angleUnit：関節角度の単位を指定
    - name：モデル名を指定

ヘッダの記述 | 項目

- format
  - 『ChoreonoidBody』と記述することで，Choreonoidのモデルファイルとして認識
- formatVersion
  - 『1.0』と記述することで，バージョン番号を明示
- angleUnit
  - degreeと記述することで，関節角度を度数法で記述
    - 単位は**degree**，**radian**を指定可能
- name
  - 『モデル名』と記述することで，Choreonoidで読み込んだ際のモデル名を指定

ヘッダの記述例

❖ TurtleBot2モデルのヘッダ部（TurteBot2.body）

```
format: ChoreonoidBody
formatVersion: 1.0
angleUnit: degree
name: TurtleBot2
```

リンク

- モデルを構成している各パーツ
  - TurtleBot2の場合：
    - 本体，プレート，右ホイール，左ホイール，
      前キャスタ，後キャスタの計6つをリンクとよぶ

- ルートリンク：モデルの中心となるパーツ
  - TurtleBot2の場合：車体がルートリンク

リンクの記述

- モデルのリンク情報は，『links:』のように記述
- リンクを記述する際は，『-』の前に，**半角スペースが2つ必要**
- 各リンクの記述部分をLinkノードとよぶ
- 最初のLinkノードは，ルートリンク（ベース）となる

```
format: ChoreonoidBody
formatVersion: 1.0
angleUnit: degree
name: モデル名

links:
  -
    リンク1(ルートリンク)の記述
  -
    リンク2の記述
  -
    リンク3の記述
 . . .
```

} **Linkノード**

Linkノード

❖ Linkノードは，以下のパラメータが利用可能

キー	内容
name	リンク名
parent	親リンクの名前（nameに記述した文字列）を指定 ルートリンクの場合は不要
translation	本リンクの親リンクからの相対位置 ルートリンクの場合はモデル読み込み時のデフォルト位置 x, y, z 方向の移動を[x, y, z]で指定（単位は [m]）
rotation	本リンクの親リンクからの相対姿勢 ルートリンクの場合はモデル読み込み時のデフォルト姿勢 回転軸と回転角度を [x, y, z, θ] で指定x, y, z には, 1：“回転軸”，0：“回転軸でない”を指定可 ただし，[1, 1, 0, 30] のような記述はできない

Linkノード

キー	内容
jointType	関節タイプ fixed（固定）, free（非固定ルートリンク）, revolute（回転関節）, prismatic（直動関節）, pseudoContinousTrack（簡易無限軌道）の いずれかを指定
jointAxis	関節軸 3次元ベクトルの3要素のリストとして関節軸の向きを [x, y, z] で指定 x, y, z には，1：“回転軸(正の回転)”, 0：“回転軸でない”，-1：“回転軸（負の回転）”を指定可 また，関節軸がローカル座標の X, Y, Z 軸と一致する場合, 対応する軸の文字（X, Y, Z）でも指定可能

Linkノード

キー	内容
jointRange	関節可動範囲 最小値, 最大値の2つの値をリストで [最小角度, 最大角度] と指定
jointId	関節ID値 モデル内で重複しない0以上の任意の整数値を指定可能
jointAngle	関節の初期角度（degreeで指定）
centerOfMass	重心位置 各リンクのローカル座標で[x, y, z]と指定（単位は [m]）
mass	質量[kg]

Linkノード

キー	内容
inertia	慣性モーメント 慣性テンソルの9要素をリストとして列挙，以下で指定する [Ixx，Ixy，Ixz， Iyx，Iyy，Iyz， Izx，Izy，Izz] ※慣性モーメント：「回転の始まりにくさ」，「回転の止まりにくさ」を表す CADツールなどで妥当な値を算出可能
elements	リンクの構成要素となる子ノードを記述

リンク形状の記述

❖ リンク形状：Linkノードの elements 以下に記述

キー	内容
type	Shape
geometry	リンクの形状を記述 指定可能な形状は，Box，Cylinder，Sphere，Capsule， Cone，Extrusion，ElevationGrid，IndexedFaceSet
appearance	リンクの色やテクスチャを記述

幾何形状の記述

❖ Boxノード：直方体を記述する幾何形状ノード

キー	内容
type	box
size	直方体の縦, 横, 奥行の長さ（[x, y, z] で指定）

❖ Cylinderノード：円柱を記述する幾何形状ノード

キー	内容
type	Cylinder
radius	半径
height	高さ
bottom	true：底面あり（default）, false：底面なし
top	true：上面あり（default）, false：上面なし

幾何形状の記述

❖ Sphereノード：球を記述する幾何形状ノード

キー	内容
type	Sphere
radius	球の半径

❖ Capsuleノード：カプセルを記述する幾何形状ノード

キー	内容
type	Capsule
radius	半径
height	高さ

幾何形状の記述

❖ Coneノード：円錐を記述する幾何形状ノード

キー	内容
type	Cone
radius	底面の半径
height	高さ
bottom	true：底面あり（default），false：底面なし

elementsの記述

❖ elements：ノードを階層的に記述するためのキー

```
elements:
  -
    type: ノードタイプ名
    key1: value1
    key2: value2
    ...
  -
    type: ノードタイプ名
    key1: value1
    key2: value2
  ...
```

リスト形式での記述

```
elements:
  ノードタイプ名:
    key1: value1
    key2: value2
    ...
```

簡略化記法での記述

リンク色の記述

❖ Appearanceノード：リンクの色やテクスチャ記述するノード

キー	内容
material	物体表面の材質を記述
texture	物体表面のテクスチャを記述

リンク色の記述

❖ Materialノード：リンクの色を記述するノード

キー	内容
ambientIntensity	環境光の反射率（0.0～1.0）
diffuseColor	RGBごとの拡散反射率（物体の色）[R, G, B] で指定 （RGBそれぞれ0.0～1.0のリスト）
emissiveColor	物体自体から発光する色 [R, G, B] で指定 （RGBそれぞれ0.0～1.0のリスト）
shininess	輝度（0.0～1.0）
specularColor	鏡面反射率（光のハイライトの色）[R, G, B] で指定 （RGBそれぞれ0.0～1.0のリスト）

リンクの記述例

❖ 以下はKobukiモデルのリンク部（Kobuki.body）

```
links:
  -
    name: kobuki
    translation: [ 0.001, 0, 0.05199 ]
    jointType: free
    centerOfMass: [ 0, 0, 0 ]
    mass: 2.4
    inertia: [
      0.019995, 0,   0,
      0,   0.019995, 0,
      0,   0,   0.03675 ]
    elements:
      -
        type: Visual
        rotation: [ 1, 0, 0, -90 ]
        elements:
          Resource:
            uri: "kobuki/kobuki_description/meshes/main_body.wrl"
      -
        type: Collision
        elements:
          -
            type: Shape
            translation: [ 0, 0, -0.0053 ]
            rotation: [ 1, 0, 0, 90 ]
            geometry: { type: Cylinder, radius: 0.175, height: 0.084 }
          -
            type: Shape
            translation: [ -0.095, 0, 0.023 ]
            rotation: [ 0, 1, 0, 60 ]
            geometry: { type: Box, size: [ 0.06, 0.23, 0.05 ] }
  -
    name: wheel_left
    parent: kobuki
```

※インデントはデータの構造も規定しているので,
上記のようにインデントが揃っているところはそのまま揃えて記述

アンカーの設定

❖ 『`appearance: &BodyAppearance`』ように,
特定の箇所に名前を付けることで，同様の記述がある場合に，参照が可能

❖ 修正が発生した場合，アンカーの設定を行っている箇所を修正すれば全てに反映される

エイリアスの利用

❖ アンカーを参照している箇所は『エイリアス』とよぶ

❖ エイリアスで参照することで，残りの記述を省略できる

❖ エイリアスとして参照するには，
以下のように，アンカーで設定した名前の前に『*』を付ける

❖ 例：『**appearance: *BodyAppearance**』

RigidBodyの記述

❖ RigidBodyノード：リンクの剛体パラメータを定義するノード

キー	内容
type	RigidBody
centerOfMass	重心位置
mass	質量[kg]
inertia	慣性モーメント（慣性テンソルの9要素をリストとして列挙）
elements	子ノードでリンクの形状やセンサーなどを記述

RigidBodyの記述例

❖ Kobuki.bodyのRigidBody使用箇所（一部省略）

```
links:
  -
    name: kobuki
  -
    name: wheel_left
    parent: kobuki
    translation: [ 0, 0.115, -0.02695 ]
    jointType: revolute
    jointAxis: Y
    jointRange: [ -10, 45 ]
    elements: &WHEEL
      -
        type: RigidBody
        centerOfMass: [ 0, 0, 0 ]
        mass: 0.076
        inertia: [ 0.001, 0, 0,
                   0, 0.001, 0,
                   0, 0, 0.001 ]
      -
        type: Visual
        rotation: [ 1, 0, 0, 90 ]
        elements:
          Resource:
            uri: "kobuki/kobuki_description/meshes/wheel.wrl"
      -
        type: Collision
        elements:
          Shape:
            geometry: { type: Cylinder, radius: 0.0352, height: 0.0206 }
  -
    name: wheel_right
    parent: kobuki
    translation: [ 0, -0.115, -0.027 ]
    jointType: revolute
    jointAxis: Y
    centerOfMass: [ 0, 0, 0 ]
    elements: *WHEEL
```

Transformの記述

- Transformノード
  - 配下のノードを平行移動・回転・拡大縮小するノード

キー	内容
type	Transform
translation	位置のオフセット
rotation	姿勢のオフセット
scale	サイズの拡大・縮小を [x, y, z] で指定 x, y, z には拡大・縮小率を指定（デフォルトサイズは 1.0）
elements	変換を受ける子ノードを記述

Transformの記述例

❖ Kobuki.bodyのTransform使用箇所（一部省略）

```
links:
  -
    name: kobuki
  -
    name: wheel_left
    parent: kobuki
  -
    name: wheel_right
    parent: kobuki
  -
    name: caster_front
    parent: kobuki
  -
    name: caster_back
    parent: kobuki
```

```
  -
    name: sensors
    parent: kobuki
    jointType: fixed
    elements:
      -
        type: Transform
        translation: [ 0.056, 0.062, 0.0202 ]
        mass: 0.001
        inertia: [ 0.0001,   0,   0,
                   0, 0.000001,   0,
                   0,   0,   0.0001 ]
        elements:
          -
            type: RateGyroSensor
            name: Gyro_sensor
            id: 0
```

デバイスの記述 | 共通項目

❖ 各種デバイスでの共通設定項目

キー	内容
name	デバイスの名前
id	デバイスのID
translation	ローカル座標系の位置を指定
rotation	ローカル座標系の姿勢を [x, y, z, θ] で指定（ベクトル [x, y, z] の周りを θ 回転）

デバイスの記述 | カメラ

❖ Cameraノード：視覚センサを定義するノード

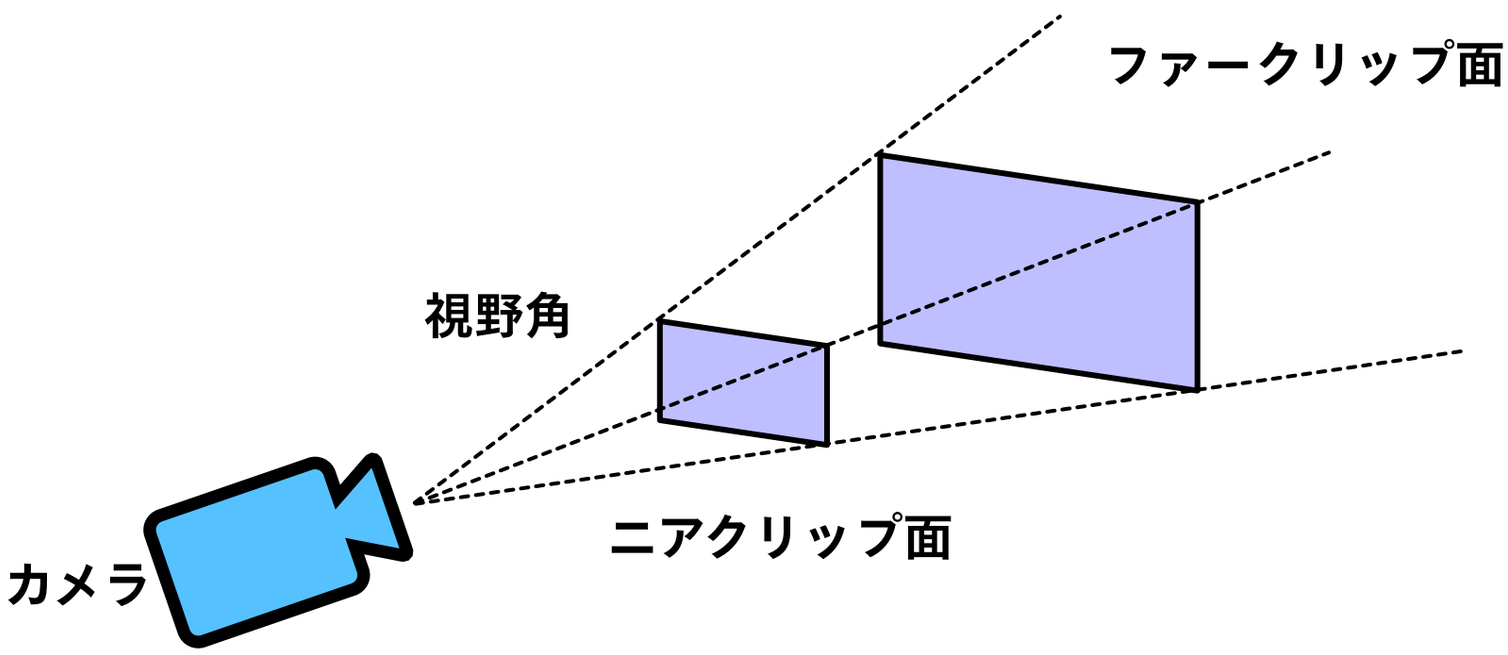
キー	内容
type	Camera
format	センサから取得する情報の種類を指定 COLOR：色情報を取得 DEPTH：深さ情報を取得 COLOR_DEPTH：色と深さ情報を取得 POINT_CLOUD：3次元点群を取得 COLOR_POINT_CLOUD：色と3次元点群を取得
lensType	レンズの種類を指定 NORMAL：通常レンズ（default） FISHEYE：魚眼レンズ DUAL_FISHEYE：全方位カメラ
on	true/falseでカメラのON/OFFを指定

デバイスの記述 | カメラ

キー	内容
width	画像の幅
height	画像の高さ（lensType=“FISHEYE”, “DUAL_FISHEYE”の場合は, widthの値から自動で決定）
fieldOfView	カメラの視野角度（lensType=“DUAL_FISHEYE”の場合は, 指定不可）
nearClipDistance	視点からニアクリップ面までの距離
farClipDistance	視点からファークリップ面までの距離
frameRate	カメラが毎秒何枚の画像を出力するか

デバイスの記述 | 補足

- ニアクリップ面/ファークリップ面とは,
  カメラ画像を描画する時の視点から,
  最も近い距離と最も遠い距離の面のこと
- 視点の姿勢は以下のように定義されている
  - 視線前方向：**ローカル座標系でZ軸の負の向き**
  - 視線上方向：**ローカル座標系でY軸の正の向き**



デバイスの記述例

❖ TurtleBot2.bodyのカメラデバイス部（一部省略）

```
links:
  -
    name: TurtleBot
    type: SubBody
    uri: "Kobuki.body"
    jointType: free
  -
    name: hexagons_plate
    type: SubBody
    parent: kobuki
    translation: [ -0.002, 0.002, -0.052 ]
    uri: "HexagonsPlate.body"
    jointType: fixed
  -
    name: Camera
    parent: kobuki
    translation: [ 0.17, 0, 0.2 ]
    rotation: [ [ 0, 0, 1, 90 ], [ 1, 0, 0, 50 ] ]
```

```
    jointType: fixed
    elements:
      -
        type: Camera
        name: LineTrace
        rotation: [ [ 0, 0, 1, 180 ], [ 1, 0, 0, 90 ] ]
        format: COLOR_DEPTH
        lensType: Normal
        on: true
        width: 320
        height: 240
        fieldOfView: 80
        nearClipDistance: 0.02
        farClipDistance: 1.5
        frameRate: 30
        id: 5
```

メッシュの読み込み

- Resourceノード
  - リンクの形状にCADや3Dモデリングツールで作成したメッシュを読み込むノード

キー	内容
type	Resource
uri	リンク形状のメッシュファイルのパス
node	メッシュファイル内の特定のノードのみを読み込む場合に，ノード名を指定

表示用モデルの記述

❖ Visualノード：物理計算を行わず表示のみを行うノード

キー	内容
type	Visual
translation	ローカル座標系の位置を指定
rotation	ローカル座標系の姿勢を [x, y, z, θ] で指定 （ベクトル [x, y, z] の周りをθ回転）
elements	子ノードでリンクの形状やセンサなどを記述

❖ CADやモデリングツールで作成したメッシュ，
幾何形状の表示のみを行う

❖ 干渉用モデルとセットで記述

❖ 表示用モデルとして幾何形状の記述を行う場合もある

干渉モデルの記述

- Collisionノード
  - 形状の表示は行わず物理計算にのみ使用するノード

キー	内容
type	Collision
translation	ローカル座標系の位置を指定
rotation	ローカル座標系の姿勢を [x, y, z, θ] で指定 （ベクトル [x, y, z] の周りをθ回転）
elements	子ノードでリンクの形状を記述

- メッシュと同様の形状となるように幾何形状のみで構成
- 表示用と干渉用モデルを分けることにより，
  実時間に近いシミュレーションが可能

表示用と干渉用の記述例

❖ Kobuki.bodyの本体リンク部（一部省略）

```
links:
  -
    name: kobuki
    translation: [ 0.001, 0, 0.05199 ]
    jointType: free
    centerOfMass: [ 0, 0, 0 ]
    mass: 2.4
    inertia: [ 0.019995,   0,   0,
               0,   0.019995,   0,
               0,   0,   0.03675 ]
    elements:
      -
        type: Visual
        rotation: [ 1, 0, 0, -90 ]
        elements:
          Resource:
            uri: "kobuki/kobuki_description/meshes/main_body.wrl"
      -
        type: Collision
        elements:
          -
            type: Shape
            translation: [ 0, 0, -0.0053 ]
            rotation: [ 1, 0, 0, 90 ]
            geometry: { type: Cylinder, radius: 0.175, height: 0.084 }
          -
            type: Shape
            translation: [ -0.095, 0, 0.023 ]
            rotation: [ 0, 1, 0, 60 ]
            geometry: { type: Box, size: [ 0.06, 0.23, 0.05 ] }
  -
    name: wheel_left
    parent: kobuki
```