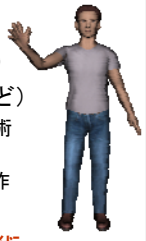
コンピュータグラフィックス特論 II

第10回　キャラクタアニメーション（1）

九州工業大学　尾下 真樹

キャラクタ・アニメーション

- CGにより表現されたキャラクタのアニメーションを実現するための技術
- キャラクタ・アニメーションの用途
  - オフライン・アニメーション（映画など）
  - オンライン・アニメーション（ゲームなど）
    - どちらの用途でも使われる基本的な技術は同じ（データ量や詳細度が異なる）
    - 後者の用途では、インタラクティブな動作実現のための工夫が必要になる
- 人体モデル・動作データの処理技術



全体の内容

- キャラクタ・アニメーションの基礎
- 骨格モデル・姿勢・動作の表現
- 動作データの作成
- 運動学
- 姿勢・動作ブレンディング
- 応用技術

今日の内容

- キャラクタ・アニメーションの基礎
- 骨格モデル・姿勢・動作の表現
  - 骨格モデルの表現
  - 姿勢・動作の表現
  - ワンスキンモデル
  - BVH動作データの読み込みと描画
- 動作データの作成
  - モーションキャプチャ
  - キーフレームアニメーション

キャラクタ・アニメーションの基礎

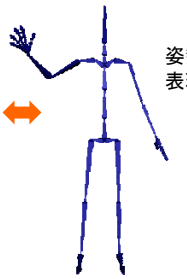
人間の各要素の表現

- 人体の表現
  - 身体の表現
    - 全身の骨格・形状の表現
  - 顔の表現
    - 細かい表情変化を表現するためには体とは別のモデルが必要
- 付属物の表現
  - 髪の毛や衣服など
    - シミュレーションによる動きの計算

人体モデルの表現

形状モデル（ポリゴンモデル） 骨格モデル（多関節体）

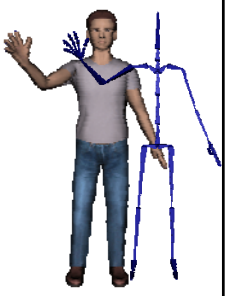
描画用



姿勢・動作の表現・処理用

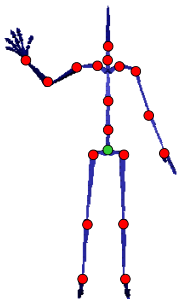
骨格モデルの表現

- 人間を多関節体として扱う
  - 人間の骨格をモデル化するのに、大体40～200くらいの自由度が必要になる
    - 基本的な関節だけで 40程度
    - 手の指や足の指なども入れると 200自由度くらい必要
  - 全関節の角度により人間の姿勢を表せる
  - 人体の形状モデルは、骨格の動きに応じて変形



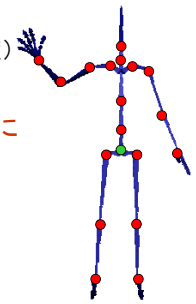
骨格モデルの表現

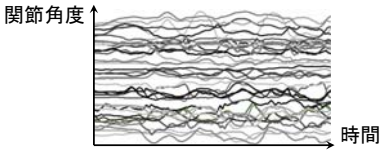
- 多関節体モデルによる表現
  - 複数の体節（リンク）が関節で接続されたモデル
  - 体節
    - 複数の関節が接続されており、体節の長さや体節内での関節の接続位置は固定
  - 関節
    - 2つの体節の間を接続
    - 関節の回転により姿勢が変化する



姿勢・動作データの表現

- 多関節体の姿勢の表現
  - 各関節の回転角度（40~自由度）
  - 腰の位置・向き（6自由度）
- 関節角度（姿勢）の時間変化により多関節体の動作を表現





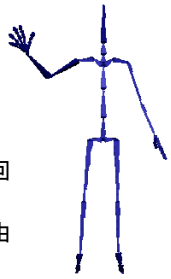
動作データの例

関節の回転の表現

- キーフレームアニメーションの回に学習した、3種類の向きの表現方法のどれかを使用
  - オイラー角による表現
  - 回転行列による表現
  - 四元数による表現
    - これらの表現は互いに変換可能なので、必要に応じて変換できる
    - 複数の表現を持たせるようにしても良い（どちらにしてもレンダリング時には回転行列が必要）

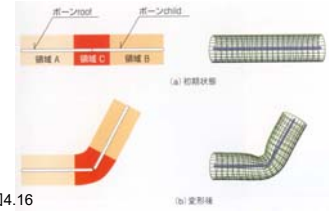
関節の自由度の種類

- 3自由度関節
  - 人間の大部分の関節は3自由度
- 1自由度関節
  - ひじやひざなどの関節
    - 2自由度以上も多少は回転可能
  - オイラー角表現の場合は1つの回転角度で表現できる
  - 回転行列・四元数の場合は1自由度になるように制約を適用
    - 全関節を3自由度とするならば省略可



ワンスキンモデル

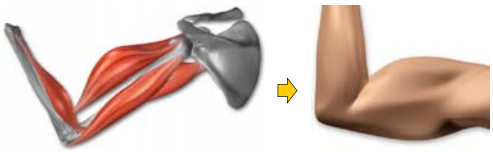
- 人間の形状を全身で1つのポリゴンモデルとして作成
- 骨格モデルの変形に応じてポリゴンモデルの各頂点を移動



参考書 図4.16

より高度な変形モデル(1)

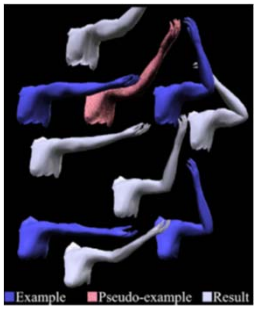
- 筋肉モデルにもとづく変形
  - 人間の筋肉をモデリング
  - 骨格の動きに応じて筋肉を伸縮
  - 筋肉の動きに応じて皮膚を変形
    - ※ ワンスキンモデルよりもリアルな変形を実現できる



[Softimage XSI]

より高度な変形モデル(2)

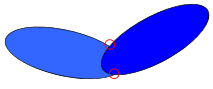
- 形状補間
  - あらかじめ入力された複数のサンプル形状を適切な重みでブレンドすることで、変形を実現
  - 姿勢に応じて、適切なブレンドの重みを計算する必要がある



[Sloan 01]

剛体によるキャラクタの表現

- 各リンクを幾何形状モデルとして表す方法
  - 関節部分ではポリゴンがめり込むが、全体としてはつながって見える
    - 正確な陰面消去が必要
    - 境界が不自然になる
  - 昔はよく使われていた
  - ロボットなどの表現には有効



動作データの表現方法

- 一定間隔データ
  - BVH形式のように、一定間隔ごとの各フレームの姿勢データを持つ方法
  - 姿勢データの配列により表現できる
- キーフレームデータ
  - キーフレームの姿勢データのみを持ち、中間の姿勢は補間によって求める方法
    - ルートの位置・回転の補間、関節の回転の補間の方法については以前の講義で学習した通り
  - (時刻、姿勢データ)の組の配列により表現
    - 各関節ごとの別のキー時刻を持つ方法もある

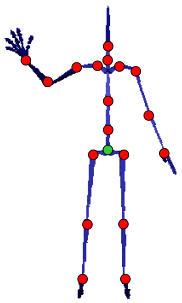
今日の内容

- キャラクタ・アニメーションの基礎
- 骨格モデル・姿勢・動作の表現
  - 骨格モデルの表現
  - 姿勢・動作の表現
  - ワンスキンモデル
  - BVH動作データの読み込みと描画
- 動作データの作成
  - モーションキャプチャ
  - キーフレームアニメーション

骨格モデル・姿勢・動作の表現

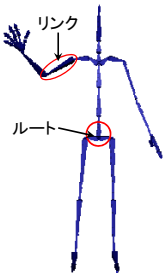
骨格モデルの表現

- 多関節体モデルによる表現
  - 複数の体節(リンク)が関節で接続されたモデル
  - 体節
    - 複数の関節が接続されており、体節の長さや体節内での関節の接続位置は固定
  - 関節
    - 2つの体節の間を接続
    - 関節の回転により姿勢が変化する



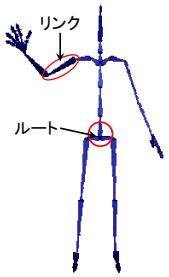
骨格モデルの表現方法

- 多関節体の骨格・姿勢の表現
  - リンク(体節・関節)の集合として表現
  - ルートとなるリンクを元として、ツリー状に多数のリンクが接続
- 具体的にどのようなデータ構造により表現するかは、いくつかの方法がある



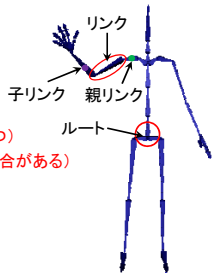
骨格モデルの表現方法(1)

- 最も単純な表現方法
  - 各リンクを表すクラスを定義
  - リンクに含まれる情報
    - 骨格情報(固定情報)
      - 隣接リンク
      - 隣接リンクとの相対位置
      - (スキニング情報)
    - 姿勢情報(動作によって変化)
      - 親リンクに対する相対回転
      - ルートの位置・向き(ルートリンクのみ)



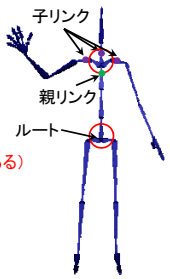
骨格モデルの表現方法(1)

- 最も単純な表現方法
  - 各リンクを表すクラスを定義
  - リンクに含まれる情報
    - 骨格情報(固定情報)
      - 隣接リンク
        - » 親リンク(ルート側、常に一つ)
        - » 子リンク(末端側、複数の場合がある)
      - 隣接リンクとの相対位置
      - (スキニング情報)
    - 姿勢情報(動作によって変化)
      - 親リンクに対する相対回転
      - ルートの位置・向き(ルートリンクのみ)



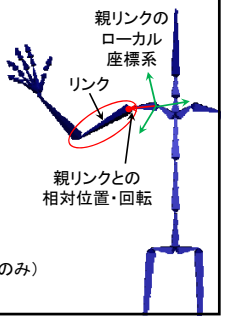
骨格モデルの表現方法(1)

- 最も単純な表現方法
  - 各リンクを表すクラスを定義
  - リンクに含まれる情報
    - 骨格情報(固定情報)
      - 隣接リンク
        - » 親リンク(ルート側、常に一つ)
        - » 子リンク(末端側、複数の場合がある)
      - 隣接リンクとの相対位置
      - (スキニング情報)
    - 姿勢情報(動作によって変化)
      - 親リンクに対する相対回転
      - ルートの位置・向き(ルートリンクのみ)



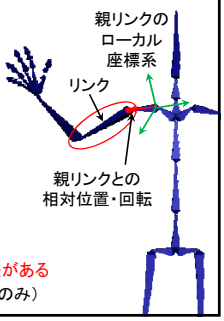
骨格モデルの表現方法(1)

- 最も単純な表現方法
  - 各リンクを表すクラスを定義
  - リンクに含まれる情報
    - 骨格情報(固定情報)
      - 隣接リンク
      - 隣接リンクとの相対位置
        - » 親リンクの座標系での位置
      - (スキニング情報)
    - 姿勢情報(動作によって変化)
      - 親リンクに対する相対回転
      - ルートの位置・向き(ルートリンクのみ)



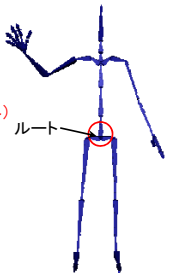
骨格モデルの表現方法(1)

- 最も単純な表現方法
  - 各リンクを表すクラスを定義
  - リンクに含まれる情報
    - 骨格情報(固定情報)
      - 隣接リンク
      - 隣接リンクとの相対位置
      - (回転自由度・軸の情報)
      - (スキニング情報)
    - 姿勢情報(動作によって変化)
      - 親リンクに対する相対回転
        - » 回転の表現には複数の方法がある
      - ルートの位置・向き(ルートリンクのみ)



骨格モデルの表現方法(1)

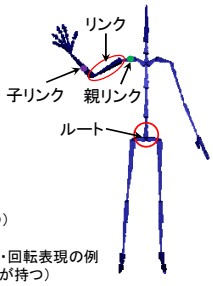
- 最も単純な表現方法
  - 各リンクを表すクラスを定義
  - リンクに含まれる情報
    - 骨格情報(固定情報)
      - 隣接リンク(ルートリンクは子リンクのみ)
      - 隣接リンクとの相対位置
        (ルートリンクは持たない)
      - (スキニング情報)
    - 姿勢情報(動作によって変化)
      - 親リンクに対する相対回転
        (ルートリンクは持たない)
      - ルートの位置・向き(ルートリンクのみ)



骨格モデルの表現方法(1)

```
// 多関節体の各リンクを表す構造体
struct  Link
{
    // 親リンク(常に一つ、ルートはなし)
    Link *              parent;
    // 子リンク(複数になることがある)
    vector< Link * >    children;
    // 親リンクからの接続位置(ローカル座標系)
    Point3f             offset;
    // ルートリンクかどうかのフラグ
    bool                is_root;

    // ルートリンクの位置・向き(ルートのみが持つ)
    Point3f     root_pos;
    Matrix3f    root_ori;  // 回転行列による向き・回転表現の例
    // 親リンクに対する相対的な回転(ルート以外が持つ)
    Matrix3f    link_rot;  // 回転行列による向き・回転表現の例
};
```



骨格モデルの表現方法(1)

- 多関節体の骨格・姿勢の表現
  - リンクの集合(ツリー構造)により表現
    - どちらの方法(あるいは組み合わせ)でも可能

```
// 多関節体の骨格・姿勢を表す構造体
struct  SkeletonAndPosture
{
    // 全リンクの配列(0番目の要素をルートリンクとする)
    vector< Link * >  links;
};
```

```
// 多関節体の骨格・姿勢を表す構造体
struct  SkeletonAndPosture
{
    // ルートリンク(リンクのツリー構造により表現)
    Link *              root_link;
};
```

骨格モデルの表現方法(2)

- 骨格情報と姿勢情報を分ける方法
  - 骨格情報を表すクラス(基本的に固定の情報)
  - 全身の姿勢を表すクラス
    - 腰の位置・向き(6自由度)
    - 各関節の回転(n自由度)
  - こちらの方法の方が使いやすい
    - 同一骨格の複数の姿勢データが表せる
    - 同じキャラクタが複数登場するときも骨格データは1つで済む

骨格モデルの表現方法(2)

```
// 多関節体の各リンクを表す構造体(骨格情報のみ)
struct  Link
{
    // 親リンク(常に一つ、ルートはなし)
    Link *              parent;
    // 子リンク(複数になることがある)
    vector< Link * >    children;
    // 親リンクからの接続位置(ローカル座標系)
    Point3f             offset;
};

// 多関節体の骨格を表す構造体
struct  Skeleton
{
    // 全リンクの配列(0番目の要素をルートリンクとする)
    vector< Link * >  links;
};
```

骨格モデルの表現方法(2)

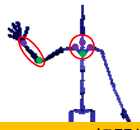
- 骨格情報と姿勢情報を分ける
  - 姿勢情報から骨格情報を参照

```
// 多関節体の姿勢を表す構造体
struct  Posture
{
    Skeleton *  body;
    Point3f     root_pos;       // ルートの位置
    Matrix3f    root_ori;       // ルートの向き(回転行列表現)
    Matrix3f *  link_rotations; // 各リンクの相対回転(回転行列表現)
                                // [リンク番号] リンク数分の配列
};
```

骨格モデルの表現方法(3)

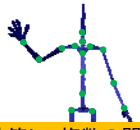
- 骨格情報の中で、関節・体節を分ける

関節・体節をまとめる



体節に、一つの親関節と複数の子関節の情報を含める(体節を順番に辿るとき、方向により異なる処理が必要)

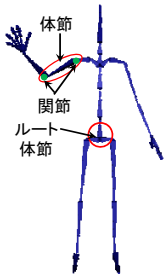
関節・体節を分ける



体節に、複数の関節へのリンクを含める(体節に順番に辿るとき、どの方向も同じ処理が可能)

骨格モデルの表現方法(3)

- 骨格情報の中で、関節・体節を分ける
  - 体節
    - 複数の関節と接続
    - 各関節の接続位置
      - 体節のローカル座標系
  - 関節
    - 2つの体節の間を接続
      - ルート側・末端側の体節



骨格モデルの表現方法(3)

- 骨格情報の中で、体節と関節を分ける

```
// 多関節体の体節を表す構造体
struct  Segment
{
    // 接続関節
    vector< Joint * >     joints;
    // 各関節の接続位置(体節のローカル座標系)
    vector< Point3f >     joint_positions;
};

// 多関節体の関節を表す構造体
struct  Joint
{
    // 接続体節
    Segment *             segments[ 2 ];
};
```

```
// 多関節体の骨格を表す構造体
struct  Skeleton
{
    // 体節・関節の配列
    vector< Segment * >  segments;
    vector< Joint * >    joints;
};
```

骨格モデルの表現方法(3)

- 骨格情報と姿勢情報を分ける

```
// 多関節体の姿勢を表す構造体
struct  Posture
{
   Skeleton *  body;
   Point3f      root_pos;        // ルートの位置
   Matrix3f     root_ori;        // ルートの向き(回転行列表現)
   Matrix3f *   joint_rotations; // 各関節の回転(回転行列表現)
                                 // [リンク番号] リンク数分の配列
};
```

骨格モデルの表現方法(3)

- 関節の回転やルートの向きの表現方法
  - 回転行列(3×3行列)以外の表現方法もある
    - 四元数の場合、リンク数分の配列(回転行列と同様)
    - オイラー角の場合、全リンクの自由度の総和分の配列
      - 自由度が異なる関節を混在させる場合、やや複雑になる
- 関節回転の可動範囲の定義(詳細は省略)
  - 可動範囲を定義して姿勢を制限する方法もある
- 筋骨格モデル(詳細は省略)
  - 動作解析・生成のため筋力を考慮する方法もある

骨格モデルの表現方法のまとめ

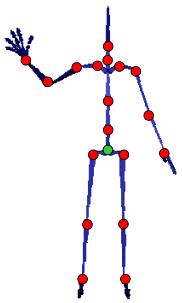
- 骨格情報と姿勢情報を分ける
- 骨格情報の中で、体節と関節を分ける

```
// 多関節体の体節を表す構造体
struct  Segment

// 多関節体の関節を表す構造体
struct  Joint

// 多関節体の骨格を表す構造体
struct  Skeleton

// 多関節体の姿勢を表す構造体
struct  Posture
```



ワンスキンモデルの作成

- ワンスキンモデルに必要な情報
  - 骨格構造の情報
  - 全身の幾何形状データ
  - 骨格構造の各リンクから幾何形状の各頂点へのウェイト
    - m×n の行列データ(リンク数m、頂点数n)
- 通常はアニメーションソフトを使って作成したモデルを利用

ワンスキンモデルの出力

- 一般的に仕様が公開されているワンスキンモデルのファイル形式は少ないので、適当な独自形式を使うのが一般的
  - FBX、Collada、X などは、ワンスキンモデルも利用可能
- 各情報を個別に出力して読み込むことも可能
  - 骨格構造＋初期姿勢の情報
    → BVH形式で出力できる
  - 全身の幾何形状データ
    → Obj形式などで出力できる
  - 骨格構造の各リンクから幾何形状の各頂点へのウェイト
    → ソフトによっては独自のテキスト形式で出力可能

ワンスキンモデルの表現方法

- 変形のためのウェイト情報は、行列(2次元配列)により表現できる

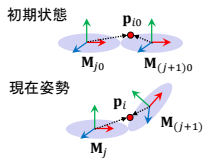
```
// ワンスキンモデルを表す構造体
struct  OneSkinModel
{
   // 骨格情報
   Skeleton *    skeleton;
   // 幾何形状情報
   Obj *         skin_shape;
   // 変形のためのウェイト情報
   float **      weights;  // [頂点番号][体節番号] の2次元配列
   // 初期姿勢での各体節の変換行列の逆行列
   Matrix4f *    init_seg_frames;  // [体節番号]
};
```

初期状態の姿勢から計算

ワンスキンモデルの変形方法

- 各頂点の位置を、各体節の変換行列とウェイトにもとづいて計算

$$\mathbf{p}_i = w_{ij} \sum_j \mathbf{M}_j \, \mathbf{M}_{j0}^{-1} \, \mathbf{p}_{i0}$$



$\mathbf{p}_i$ 各頂点の位置
$\mathbf{M}_j$ 各体節の変換行列（現在の姿勢から計算）
w_{ij} 各頂点が各体節から受けるウェイト
$\mathbf{p}_{i0}$ 初期状態での各頂点の位置
$\mathbf{M}_{j0}$ 初期状態での各体節の変換行列

ワンスキンモデルの変形処理

```
// ワンスキンモデルの変形計算
void  DeformSkinModel( const OneSkinModel * model, const Posture & posture,
                       Point3f * vertices )
{
   // 現在姿勢での各体節の変換行列を計算

   // 各頂点の位置を計算

}
```

ワンスキンモデル情報と現在姿勢を入力
各頂点の位置を計算して出力
（頂点位置の配列の先頭アドレスを引数として受け取る）

```
// ワンスキンモデルの描画
// 幾何形状モデル（Obj形状）の描画（頂点座標の配列を入力）
void  RenderDeformedObj( const Obj * obj, const Point3f * vertices )
{
   // 幾何形状モデルが持つ頂点位置の配列の代わりに、引数として渡された
   // 頂点位置の配列を使用して描画
   // ポリゴンや素材の情報は、幾何形状モデルが持つ情報を使用
}
```

デモプログラム（1）

- キャラクタの表示
  - 市販のアニメーションソフト（3ds max）で作成したキャラクタのデータを独自形式でエクスポート
  - 独自形式ファイルの読み込み
  - ワンスキンモデルの変形・表示
- 姿勢変形
  - 手先の位置・向きをマウスで操作すると、姿勢を変形
  - 逆運動学計算（後述）

動作データの表現方法

- 一定間隔データ
  - BVH形式のように、一定間隔ごとの各フレームの姿勢データを持つ方法
  - 姿勢データの配列により表現できる
- キーフレームデータ
  - キーフレームの姿勢データのみを持ち、中間の姿勢は補間によって求める方法
    - ルートの位置・回転の補間、関節の回転の補間の方法については以前の講義で学習した通り
  - （時刻、姿勢データ）の組の配列により表現
    - 各関節ごとの別のキー時刻を持つ方法もある

動作データの表現方法

- 一定間隔データとキーフレームデータ

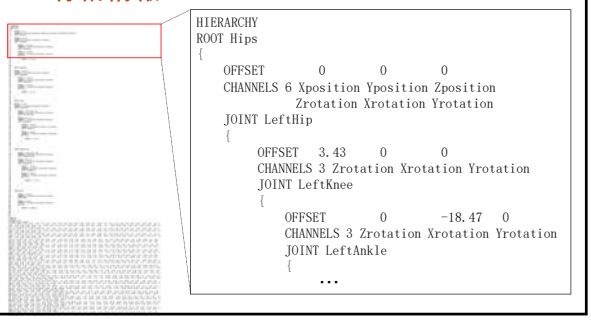
```
// 動作データ（一定間隔）
struct  Motion
{
   int          num_frames;   // 全フレーム数
   float        interval;     // フレーム間の時間間隔
   Posture *    frame_poses;  // 姿勢配列 [フレーム番号]
};
// 動作データ（キーフレーム）
struct  KeyframeMotion
{
   int          num_keyframes;  // キーフレーム数
   float *      key_times;    // 各キー時刻の配列［キーフレーム番号］
   Posture *    key_poses;    // 各キー姿勢の配列［キーフレーム番号］
};
```

モーションデータ形式の例

- 仕様が公開されているフォーマットは少ない
  - BVH・BVA、ASF-AMC、FBX（MotionBuilder）、VRML、X、COLLADA
- BVH形式
  - アスキー形式で扱いやすい
  - 骨格情報と動作情報（各時刻の姿勢）を持つ
  - 姿勢はオイラー角表現
- ASF-AMC形式
  - 骨格（ASFファイル）＋動作（AMCファイル）
  - アスキー形式、姿勢はオイラー角表現

BVH形式の例(1)

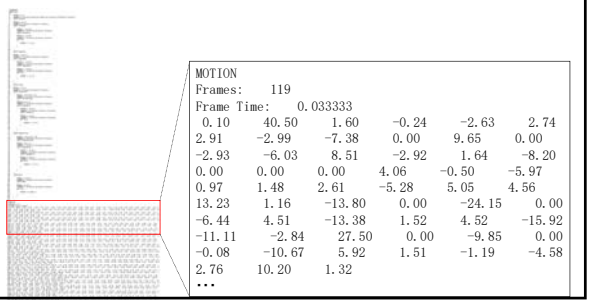
- 骨格情報



```
HIERARCHY
ROOT Hips
{
    OFFSET        0        0        0
    CHANNELS 6 Xposition Yposition Zposition
               Zrotation Xrotation Yrotation
    JOINT LeftHip
    {
        OFFSET    3.43     0        0
        CHANNELS 3 Zrotation Xrotation Yrotation
        JOINT LeftKnee
        {
            OFFSET        0        -18.47    0
            CHANNELS 3 Zrotation Xrotation Yrotation
            JOINT LeftAnkle
            {
                ...
```

BVH形式の例(2)

- 動作情報



```
MOTION
Frames:    119
Frame Time:    0.033333
 0.10    40.50    1.60    -0.24    -2.63    2.74
2.91    -2.99    -7.38    0.00    9.65    0.00
-2.93    -6.03    8.51    -2.92    1.64    -8.20
0.00    0.00    0.00    4.06    -0.50    -5.97
0.97    1.48    2.61    -5.28    5.05    4.56
13.23    1.16    -13.80    0.00    -24.15    0.00
-6.44    4.51    -13.38    1.52    4.52    -15.92
-11.11    -2.84    27.50    0.00    -9.85    0.00
-0.08    -10.67    5.92    1.51    -1.19    -4.58
2.76    10.20    1.32
...
```

デモプログラム(2)

- BVH動作データの読み込みと再生
  - BVH動作データを読み込んで再生
  - BVH動作データに記述されている骨格を表示
  - Lキーで BVHファイルを選択
    - BVHファイルは講義のページには置いていないので、各自、ネット上に公開されているものなどを探して試すこと

サンプルプログラム

- BVH動作データの読み込みと再生
- BVHクラス
  - BVHデータ構造の定義
    - なるべくBVH形式に近い形式のデータ構造を定義
    - 骨格情報 ＋ 動作情報
  - BVHファイルの読み込み
  - 読み込んだ動作データの任意のフレーム番号の姿勢を描画
- GLUT＋OpenGLを使用

BVHファイルの読み込み

- BVH形式の例
  - 詳しい仕様はネット上で探せば見つかる
- BVHファイルからの骨格情報と動作データ(姿勢の配列)の読み込み処理の例
  - サンプルプログラムを参照
  - ファイル読み込みの実現方法については、幾何形状データの読み込みの回で説明した通り
- BVHのサンプルデータ例
  - Eyes, Japan によるモーションキャプチャライブラリ http://www.mocapdata.com/

今日の内容

- キャラクタ・アニメーションの基礎
- 骨格モデル・姿勢・動作の表現
  - 骨格モデルの表現
  - 姿勢・動作の表現
  - ワンスキンモデル
  - BVH動作データの読み込みと描画
- 動作データの作成
  - モーションキャプチャ
  - キーフレームアニメーション

動作データの作成

動作データの作成

- 動作データの主な作成方法
  - キーフレームアニメーション
  - モーションキャプチャ
  - （動力学シミュレーションによる方法）
    - 3番目の方法は、やや特殊な方法なので、後述

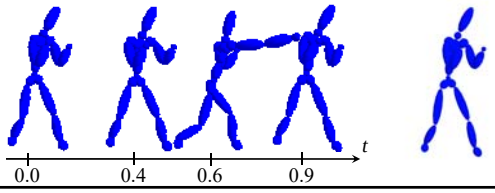
動作データの作成・表現方法

- 動作データの作成方法
  - モーションキャプチャ
  - キーフレームアニメーション
- 動作データの表現方法
  - 一定間隔データ
    - モーションキャプチャデータは通常こちらの方法で表現
  - キーフレームデータ
    - 手作業での編集にはこちらの表現の方が向いている

※ アニメーションソフトの機能で一定間隔データからキーフレームデータへの変換も可能

キーフレームアニメーション

- 動作のキーとなる姿勢を手作業で作成
  - キー姿勢の時刻・姿勢を作成
    - 各関節の回転を操作、各部位の位置を操作
- キー姿勢の間を自動的に補間して動作生成

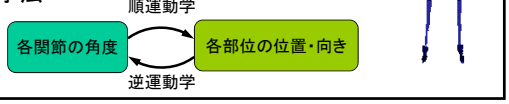


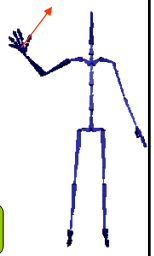
キーフレームの編集

- キーフレームにおけるキー姿勢の編集
  - 基本的にはアニメータが手作業で作成
    - かなりの時間・労力がかかる
- 基本的な姿勢の作成方法
  - 関節の回転を操作
  - 各部位の位置を操作
    - 部位の位置に合わせて関節の回転を自動計算（インバース・キネマティックス）

運動学（キネマティックス）

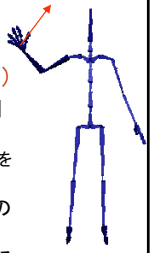
- 運動学（キネマティックス）
  - 多関節体の動きの表現の基礎となる考え方
  - 人間の姿勢は、全関節の関節角度により表現できる
  - 各関節の角度と、各部位の位置・向きの間の関係を計算するための手法





運動学（キネマティックス）

- フォワード・キネマティックス（順運動学）
  - 多関節体の関節角度から、各部位の位置・向きを計算
  - 回転・移動の変換行列の積により計算
- インバース・キネマティックス（逆運動学）
  - 指定部位の位置・向きから、多関節体の関節角度を計算
    - 手先などの移動・回転量が与えられた時、それを実現するための関節角度の変化を計算
  - 動きを指定する時、関節角度よりも、手先の位置・向きなどを使った方がやりやすい
  - ロボットアームの軌道計画等にも用いられる



モーションキャプチャ

- 人間の身体にセンサをつけて、人間の動作を計測・取得する方法
- モーションキャプチャ機器の方式
  - 光学式、慣性式、磁気式、RGB-Dカメラ式、等
  - 計測に使用するセンサーの種類の違いで、さまざまな方式がある
    - センサーから得た計測データを、動作データに変換する処理が必要となる
    - センサーの種類によっては、骨格モデルも推定可能

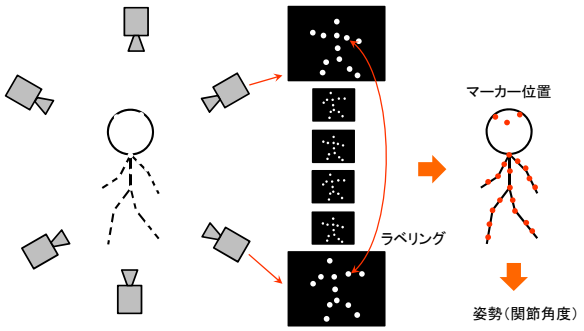
モーションキャプチャ機器の種類（1）

- 光学式
  - 現在、主に使われている方式
  - 体の各部位にマーカを付けてカメラで撮影
  - 複数のカメラから姿勢を計算
  - 姿勢の推定に時間がかかる
- 自発光学式
  - 光学式の拡張版
  - 各マーカが順番に発光
  - ラベリング処理が容易になる

Optitrack © NaturalPoint

Visualeyez © Phoenix Tech.

光学式モーションキャプチャ



モーションキャプチャ機器の種類（2）

- 慣性式
  - 加速度・ジャイロ・地磁気センサーの組み合わせにより、各部位の向きを計測
  - 小型・安価に実現可能

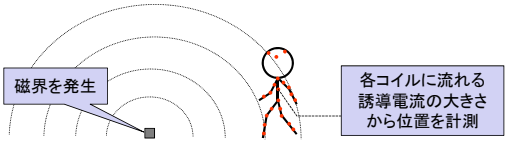
Perception Neuron © Noitom

Xsens MVN © Xsens

モーションキャプチャ機器の種類（3）

- 磁気式
  - 磁界を発生させて、体の各部位につけたセンサの位置・向きを計算
  - 高速・高精度
  - ケーブルの拘束のため動きづらい

trackSTAR © Ascension



モーションキャプチャ機器の種類(3)

- 磁気式
  - 磁界を発生させて、体の各部位につけたセンサの位置・向きを計算
  - 高速・高精度
  - ケーブルの拘束のため動きづらい

trackSTAR © Ascension

- 機械式
  - 体の各関節に回転角度を計測するための計測器を装着
  - 計測器の重さのため動きづらい

Gypsy © Meta Motion

モーションキャプチャ機器の種類(4)

- RGB-Dカメラ方式
  - 各ピクセルの奥行を計測
  - 人体部位・姿勢の推定
    - 完全な姿勢は求められない
  - 他の方式よりも精度は低い
  - Kinect が代表的
    - 事前に学習したモデルを使って、デプス画像中の各部位の位置を推定

Kinect © Microsoft

Shotten et al., "Real-Time Human Pose Recognition in Parts from a Single Depth Image", IEEE CVPR 2011.

モーションキャプチャ機器の種類(4)

- RGB-Dカメラ方式
  - 各ピクセルの奥行を計測
  - 人体部位・姿勢の推定
    - 完全な姿勢は求められない
  - 他の方式よりも精度は低い
- マーカレスカメラ方式
  - 部位の長さが既知であれば、通常のカメラのみでも姿勢推定は可能
  - RGB-Dカメラを使うよりも、さらに精度は落ちる

Kinect © Microsoft

モーションキャプチャ機器の比較

方式 (製品例)	光学式 (Optitrack)	慣性式 (Perception Neuron)	RGB-Dカメラ方式 (Kinect)
精度	高	中	低
スーツ着用	要	要	不要
専用スペース	必要	不要	不要
動作(姿勢)取得	○	○	△
骨格モデル推定	○	×	△
指の姿勢取得	△※1	○	△※2
顔の表情取得	△※1	×	△※2
最大秒間フレーム数	120	60	30
価格	数百～数千万円	数十～一千万円	数万円

※1 全身姿勢とは別にキャプチャを行う必要がある
※2 限定された状態の識別しかできない

動作データの作成方法の比較

- モーションキャプチャ
  - 人間の動きがそのまま取り込めるので、一見手軽そう
  - 実際には、ノイズや、俳優とキャラクタの骨格の違いなどのため、かなり編集が必要になる
  - そのままではキーフレームがないので編集がしづらい
- キーフレーム編集
  - ゼロから作らないといけないので大変
  - キーフレームが設定されているので、修正はしやすい
  - 人間らしい細部の動きや自然さを実現するのは難しい
- 両者の使い分けや組み合わせが必要

まとめ

- キャラクタ・アニメーションの基礎
- 骨格モデル・姿勢・動作の表現
  - 骨格モデルの表現
  - 姿勢・動作の表現
  - ワンスキンモデル
  - BVH動作データの読み込みと描画
- 動作データの作成
  - モーションキャプチャ
  - キーフレームアニメーション

今回の内容

- キャラクタ・アニメーションの基礎
- 骨格モデル・姿勢・動作の表現
- 動作データの作成
- 運動学
- 動作ブレンディング
- 応用技術

次回予告

- キャラクタ・アニメーションの基礎
- 骨格モデル・姿勢・動作の表現
- 動作データの作成
- 運動学
- 動作ブレンディング
- 応用技術