



コンピュータグラフィックス特論Ⅱ

第15回 キャラクタアニメーション(6)

九州工業大学 尾下 真樹

2021年度

今回の内容

- 動作生成・制御
 - 動作状態機械
 - 動力学を考慮した動作生成
 - 歩行動作生成
 - 深層学習による動作生成
 - 自律動作制御
 - 身体制御、動作制御、行動制御

• 講義のまとめ



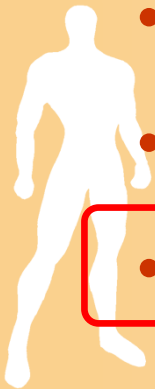
キャラクター・アニメーション

- CGにより表現された人体モデル(キャラクター)のアニメーションを実現するための技術
- キャラクター・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を実現する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

- 人体モデル(骨格・姿勢・動作)の表現
- 人体モデル・動作データの作成方法
- サンプルプログラム
- 順運動学、人体形状変形モデル
- 姿勢補間、キーフレーム動作再生、動作補間
- 動作接続・遷移、動作変形
- 逆運動学、モーションキャプチャ
- 動作生成・制御



今回の内容

- 動作生成・制御
 - 動作状態機械
 - 動力学を考慮した動作生成
 - 歩行動作生成
 - 深層学習による動作生成
 - 自律動作制御
 - 身体制御、動作制御、行動制御

• 講義のまとめ

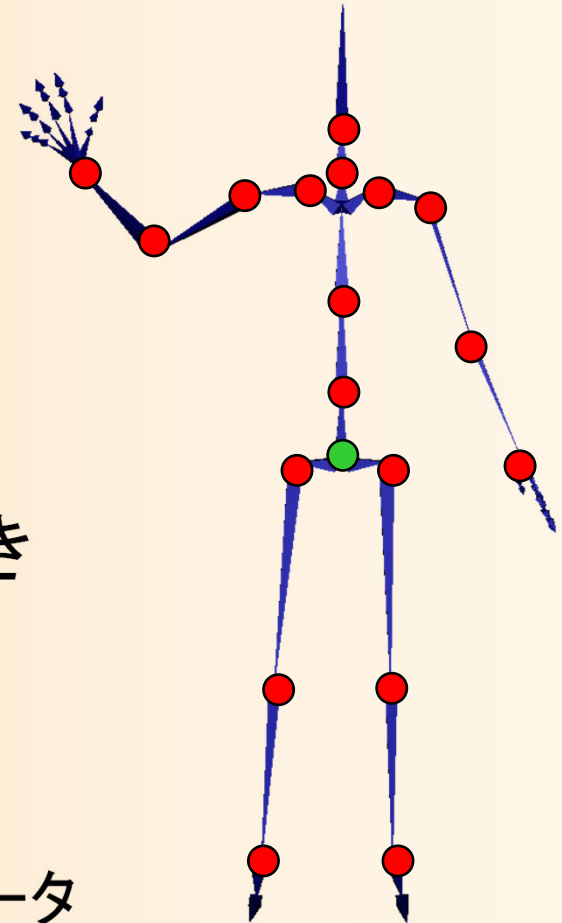




前回までの復習

骨格・姿勢・動作の表現

- 人体の骨格の表現
 - 多関節体モデルによる表現
 - 複数の体節と関節
 - 関節は2つの体節の間を接続
- 姿勢の表現
 - 全関節の回転 + 腰の位置・向き
- 動作の表現
 - 姿勢の時間変化
 - 一定間隔 or キーフレーム動作データ



基礎技術

- 人体モデル(骨格・姿勢・動作)の表現
- 順運動学
- 姿勢補間
- 動作補間
- 動作接続・遷移
- 動作変形
- 逆運動学





動作生成・制御

動作生成・制御

- 人間が作成した動作データをそのまま再生したり、変形・編集して別の動作を作成したりするのではなく、何らかのモデルにもとづいて、動作を動的に生成・制御する技術
 - 動作制御＝状況に応じた動作を動的に生成
 - 現在の技術では、人間同様の動作を生成することは難しい
 - 人間の動作制御の仕組みが、完全には分かっていないため
 - 人型ロボットの動作制御が難しい問題と同様の理由



動作制御の用途

- オフライン・アニメーション(映画など)
 - 動作生成は難しいため、モーションキャプチャやキーフレームアニメーションなどの、人間が動作データを作成する方法が用いられる
 - 一部の動作は、動作生成・制御手法により生成
- オンライン・アニメーション(ゲームなど)
 - 状況に応じた動作を行うことが求められるため、動作生成・制御手法が重要
 - 実際には、あらかじめ作成された動作データを利用して、動作生成・制御を実現



オンライン・アニメーション

- コンピュータゲームが代表例

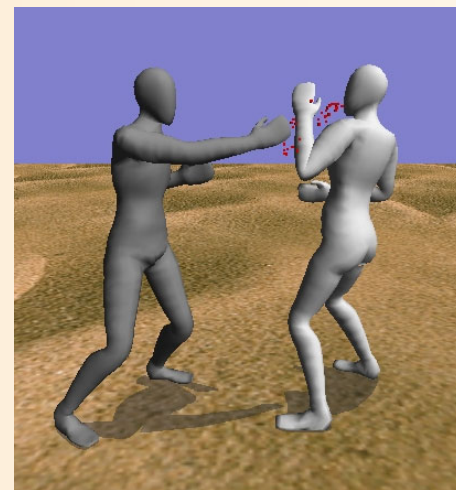
- 映画などの動画製作とは異なり、動作データを動的に生成する必要がある

- 動作の動的な生成方法

- 剛体の動作生成(車・飛行機など)
 - キー入力や単純な制御モデルによる動作
 - 物理法則を考慮する場合もある

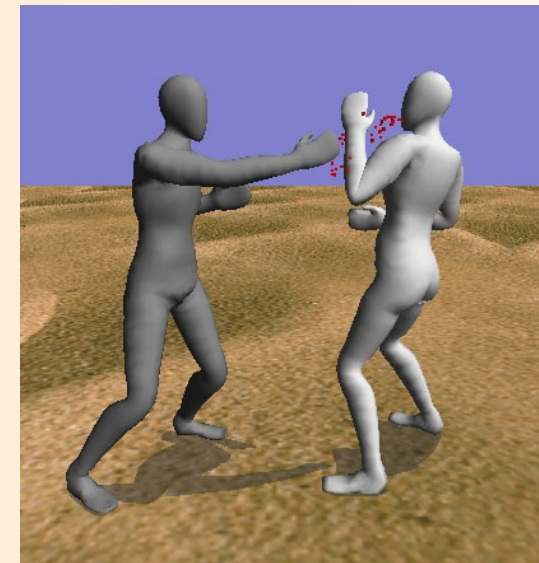
- キャラクター(多関節体)の動作生成

- 動作状態機械(次に説明)による連続動作再生
 - 必要な動作データをあらかじめ作成して用意
 - 入力などに応じて適切な動作データを順番に再生
- 物理法則は考慮されていない(不自然な動作になることがある)



オンライン・アニメーションの問題点

- 現在の主なアプローチ
 - 必要な動作データをあらかじめ全て作成しておき、ユーザの操作に応じて動作データを再生
- 問題点
 - 決められた動作の繰り返ししかできない
 - 特に衝突や外力などの力学的な影響に応じた自然な動作が困難
- 状況に応じて動的に動作を生成する技術が必要



動作生成・制御

- 動作生成・制御の技術が開発されている
- 以降、いくつかの技術について説明
 - 動作状態機械
 - 動力学を考慮した動作生成
 - 歩行動作生成
 - 自律動作制御
 - 身体制御、動作制御、行動制御





動作状態機械

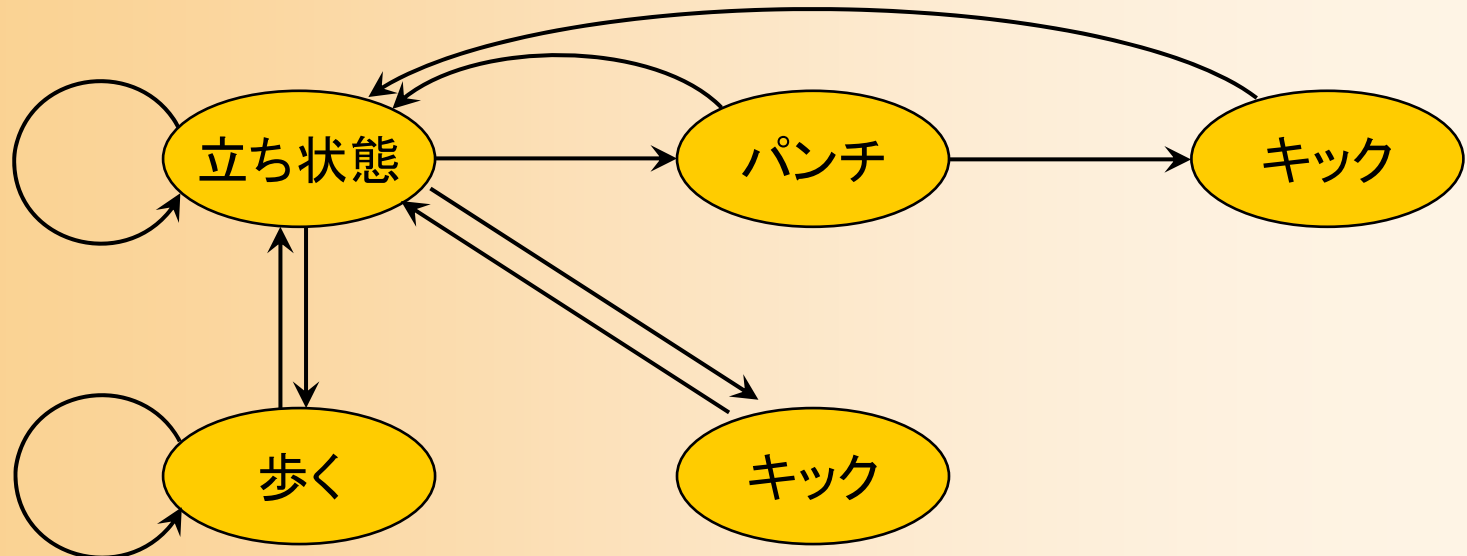
動作データの再生

- オンライン・アニメーションでの動作再生
 - 動作を連続的に行う必要がある
 - ユーザの操作や周囲の状況に応じて適切な動作を行う必要がある
- 現在のゲームなどでの一般的な方法
 - あらかじめ多数の基本的な動作データを用意
 - 適切な動作データを順番に再生することで、連続的な動作再生を実現
 - 動作データの間がなめらかにつながるように作成しておく必要がある



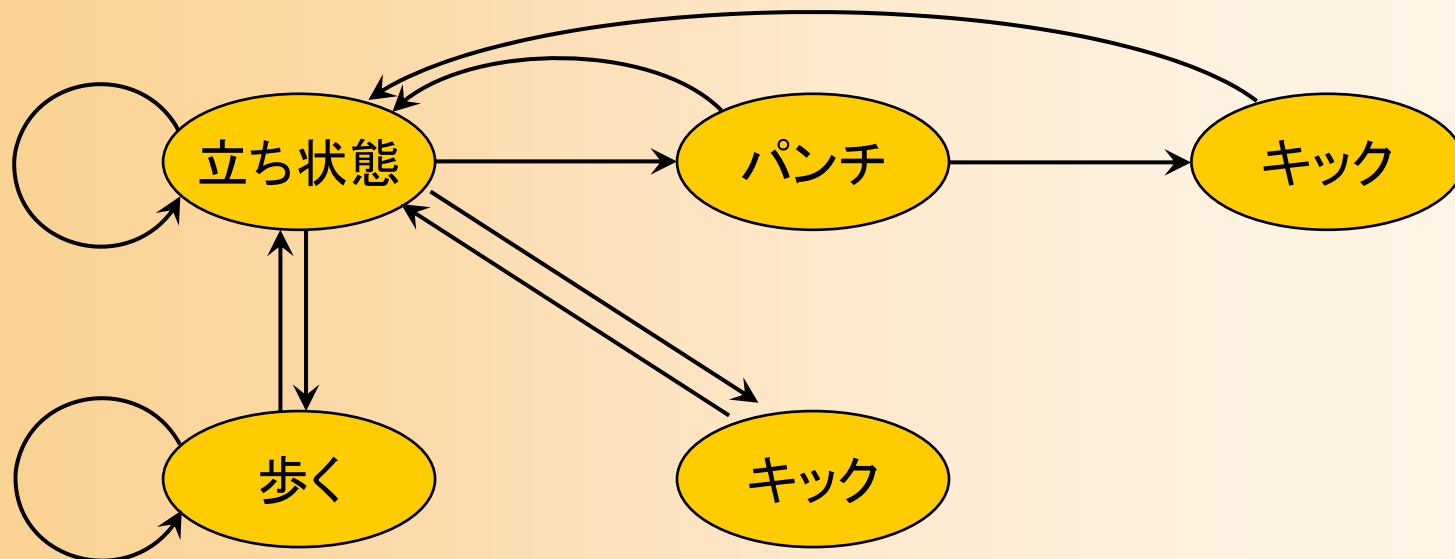
動作状態機械

- 多数の動作データと、連続して再生できる動作データ間のつながりを、グラフ構造で表現
 - 前の動作の終了姿勢と次の動作の開始姿勢が同じになるように、動作を作成
 - 各動作間の遷移のタイミングや方法を設定



動作状態機械の表現

- 動作状態機械 (State Machine)
 - データ構造としては、ノードが短い動作、エッジが前後の動作の接続を表す、有向グラフとなる
 - 動作ツリー、動作グラフなどの呼び方もある
 - 「モーショングラフ」は、別の技術(後述)を指す



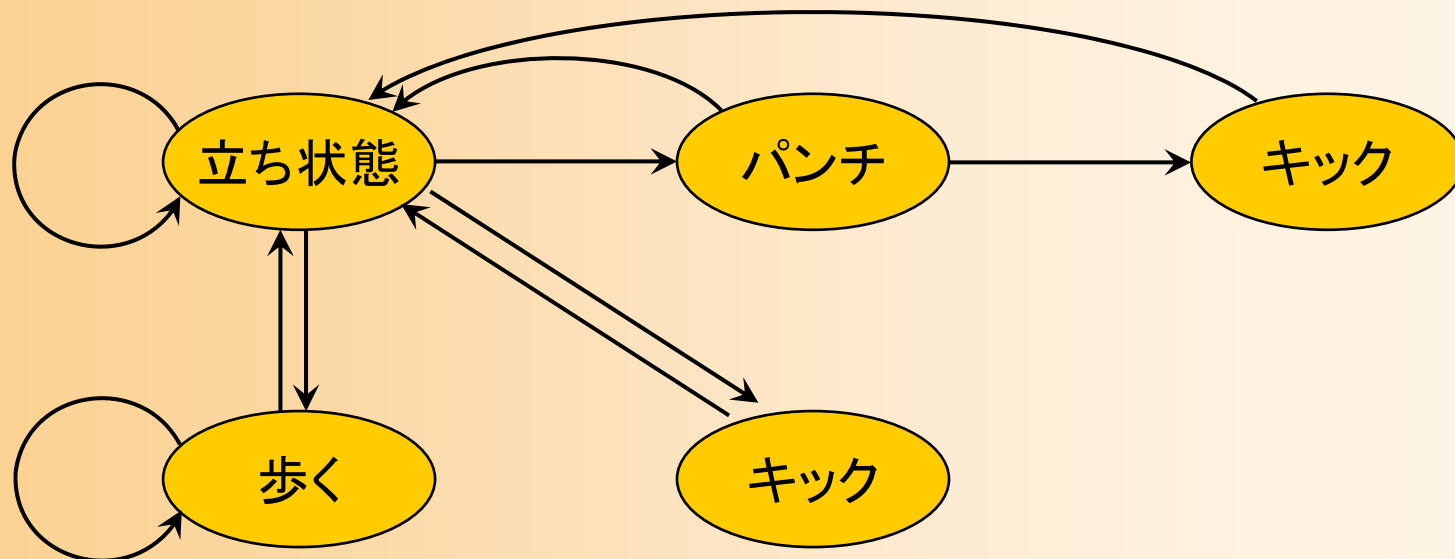
動作状態機械の作成

- それぞれの動作データは、一般的なアニメーション制作ソフトウェアを使って作成
 - 前後の動作がうまくつながるように、動作データを作成しておく
- 動作間の接続の設定は、動作状態機械の編集用のソフトウェアを使って、デザイナーが作成
 - 動作接続・遷移のパラメタ(タイミング等)を調整
 - 次の動作を選択する条件なども定義できる
 - 最近のゲームエンジンなどには、動作状態機械の編集用のソフトウェアが付属しており、利用可能
 - 例: Unity の Mechanim (Animator)



動作状態機械の利用

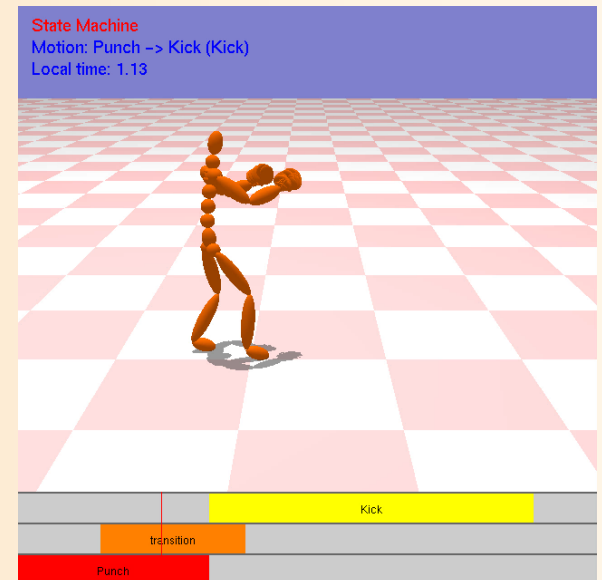
- 動作状態機械に従って、連続的に動作再生
- 次にどの動作状態に遷移するかを、利用者の操作や状況に応じて決定
 - プログラム上での処理が必要



デモプログラム

- 動作状態機械

- キー操作に応じて、次に実行する動作を選択すると、連続的に動作を再生
 - キー操作がない場合は、デフォルトの動作を再生
- 動作データや遷移情報をファイルから読み込み
- 前後の動作の切り替えには、動作接続・遷移の処理を利用
- タイムライン上に、現在と次の動作の時刻の情報を可視化



State Machine

Motion: Idle -> Idle

Local time: 0.02



動作状態機械 State Machine



動作状態機械のプログラミング(1)

- 動作状態機械を表すためのデータ構造
 - 動作状態機械、状態(動作)、遷移

```
// 状態機械クラス
class StateMachine
{
    // 骨格情報
    const Skeleton * body;

    // 全状態情報
    vector< SMState * > all_states;

    // 全遷移情報
    vector< SMTransition * > all_transitions;
}
```

```
// 状態機械の状態(動作)の情報
struct SMState
{
    // 状態に対応する動作データ
    Motion * motion;

    // 次の動作への遷移情報
    vector< SMTransition * > transitions;
}

// 状態機械の状態間(動作間)の遷移の情報
struct SMTransition
{
    // 前・後の状態(動作)
    SMState * prev_motion, * next_motion;

    // 動作接続・遷移に用いるパラメタ情報
}
```



動作状態機械のプログラミング(2)

- 動作状態機械の設定ファイル
 - 状態と遷移の情報を記述
 - 各状態の動作データは、別途作成
 - 遷移の基準部位やタイミング等は、動作データを見て決定

```
STATE Idle
MOTION fight_idle.bvh
COLOR 1.0, 1.0, 1.0
DEFAULT_NEXT_MOTION Idle
```

```
STATE Forward
MOTION fight_forward.bvh
COLOR 0.0, 1.0, 0.0
DEFAULT_NEXT_MOTION Idle
...
```

```
TRANSITION Idle Idle
BASE_SEGMENT Hips
BLEND_IN 0.25
BLEND_OUT 0.25
```

```
TRANSITION Idle Forward
BASE_SEGMENT LeftFoot
BLEND_IN 0.25
BLEND_OUT 0.25
...
```



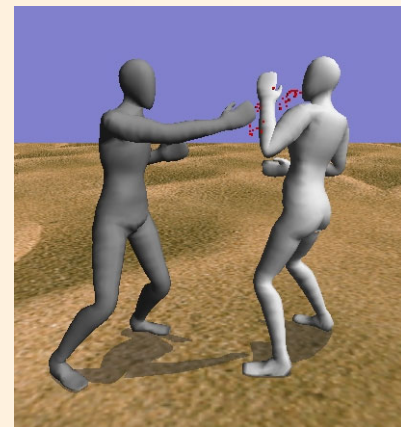
動作状態機械のプログラミング(3)

- 動作状態機械の設定ファイルの読み込み
- 次の状態(動作)の決定
 - 利用者の入力により決定
 - 入力がない場合は、自動的に決定
- 動作再生処理
 - 前後の動作間の動作遷移の処理は、動作接続・遷移のプログラムと同様
 - 動作接続・遷移の拡張
 - 強制的な動作遷移への対応
 - 現在時刻から速やかに動作遷移を開始
 - 基準部位・向きを考慮した動作接続への対応



動作状態機械による方法の問題点

- 動作状態機械の作成には多くの手間がかかる
- 一定動作の繰り返ししかできない
 - ワンパターンな動きになる
 - 特に衝突や外力などの力学的な影響に応じた自然な動作が困難
- 自律動作の実現は困難
 - 敵キャラクターや群集などを自律的に動かすためには、適切な動作ルールが必要になる





動力学を考慮した動作生成

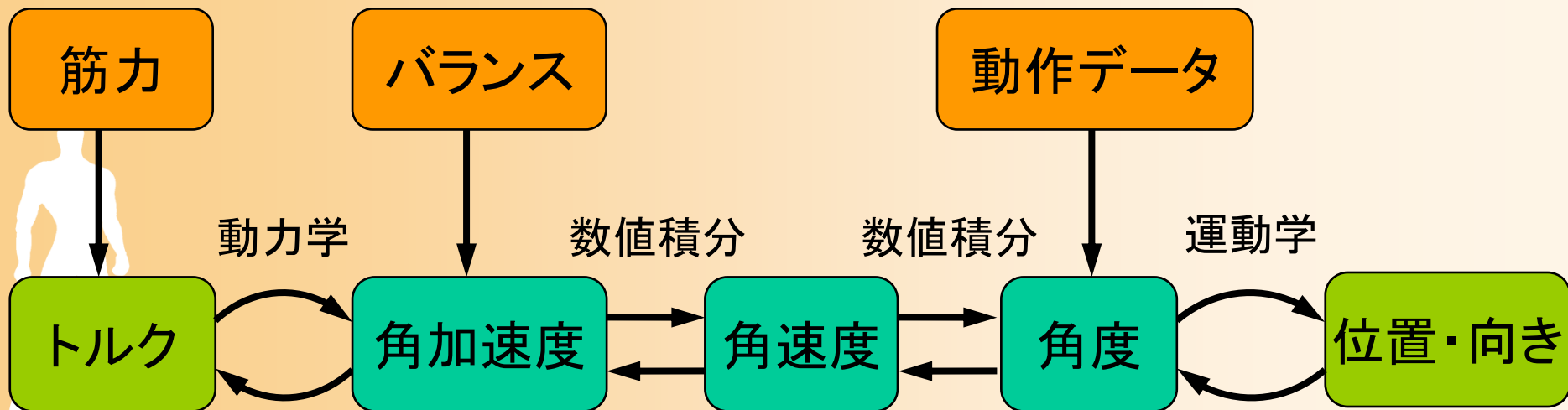
動力学を考慮したアニメーション

- 運動学(キネマティクス)
 - 多関節体の骨格構造を扱うための手法
 - 関節回転と各部位の位置・向きの関係を扱う
 - 現在は、こちらが主に使われている
 - 力学的に正しい動きが生成されることは保証されない
- 動力学(ダイナミクス)
 - 運動学に加えて、力学的な要素を考慮する手法



動力学を考慮した動作生成

- 主に2種類の方法がある
 - 動力学シミュレーションによるアプローチ
 - 動作最適化によるアプローチ
- 多関節体の動力学



動力学を考慮した動作生成

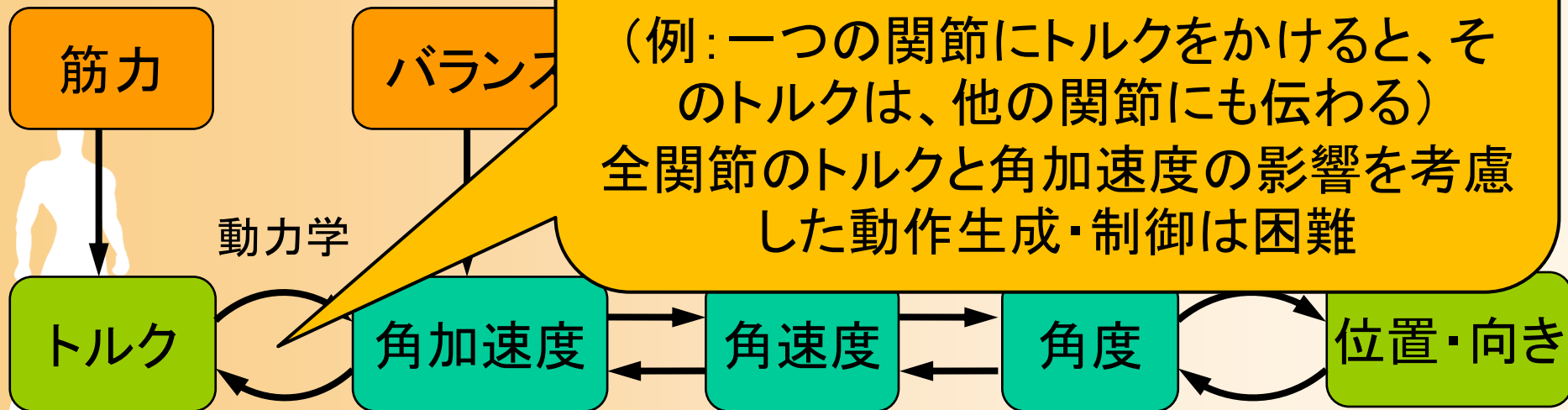
- 主に2種類の方法がある
 - 動力学シミュレーションによるアプローチ
 - 動作最適化によるアプローチ

- 多関節体の動

全ての関節のトルクが、全ての関節の角加速度に影響する

(例: 一つの関節にトルクをかけると、そのトルクは、他の関節にも伝わる)

全関節のトルクと角加速度の影響を考慮した動作生成・制御は困難



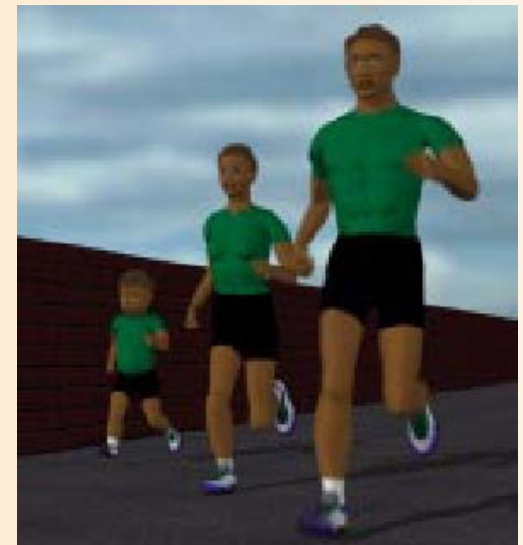
動力学シミュレーション

- コントローラ+動力学シミュレーション
 - 本物のロボットと同様に、制御理論に基づいて関節のトルクを制御するコントローラを構築
 - 動力学シミュレーションを行って動作を生成
 - 安定した動作を実現できるコントローラ的设计は困難
 - 人間らしい動作を生成することは困難

$$\tau_i = K(\theta_{i,goal} - \theta_{i,curr}) - K(\dot{\theta}_{i,goal} - \dot{\theta}_{i,curr})$$

PD制御

目標角度－現在角度に適切な係数をかけることで各関節のトルクを決定



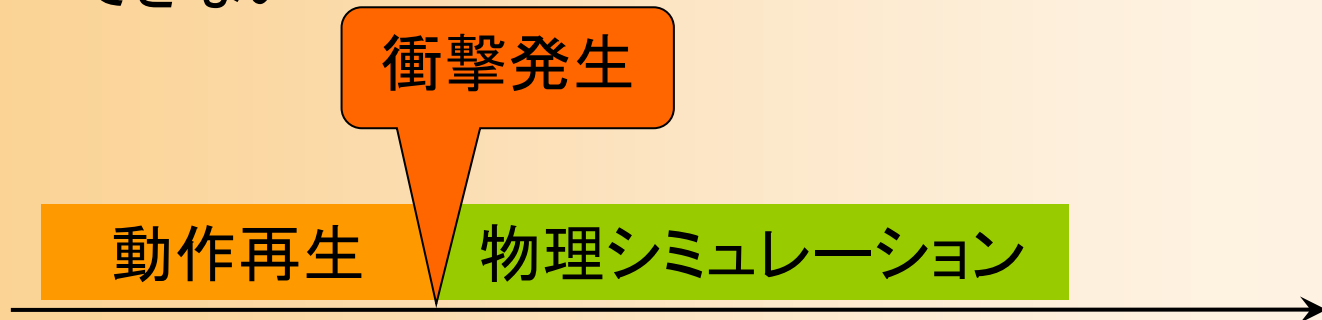
[Hodgins 97]



動力学シミュレーションの応用例

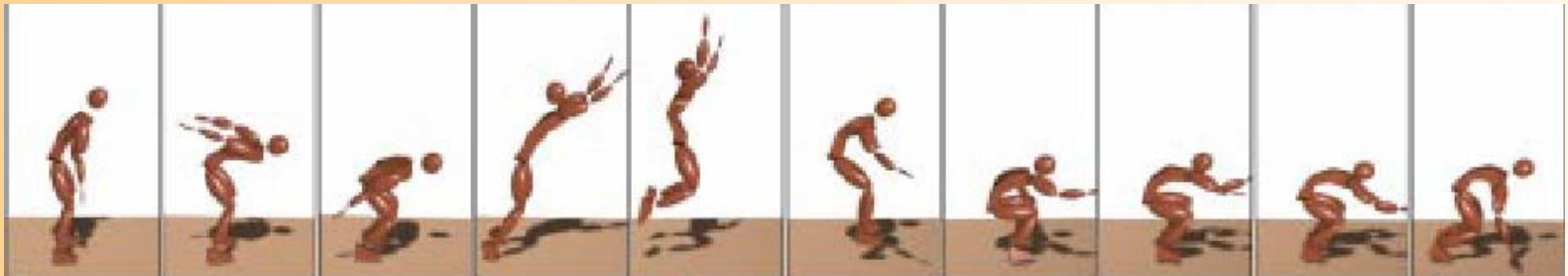
• ラグドール・シミュレーション

- 衝撃時の動作を動力学シミュレーションで生成
- 一部のコンピュータゲームで使われている
- 衝撃後のみ、動力学シミュレーションに切り替え
 - 受動的な動作のみで、能動的な動作は実現できない
 - 倒れずに途中で回復して元に戻るような動作は実現できない



動作最適化

- 動作最適化による動作変形
 - 関節トルクやバランスの不自然さを評価する目標関数 $f(\text{動作})$ を設計
 - 目標関数 $f(\text{動作})$ がなるべく小さくなるように、動作を少しずつ修正する
 - 計算に非常に多くの時間がかかる
 - 人間のような多関節体には、自由度が高すぎてそのまま適用するのは困難





步行動作生成

歩行動作の重要性

- 歩行動作
 - 人間の基本的かつ重要な動作の一つ
 - 任意の軌道に沿った歩行、地面の高低や障害物への対応、歩き方の変化、などの要素を含むため、歩行動作の生成には工夫が必要になる
 - 歩行だけでなく、移動動作全般も含む場合もある
 - 歩行、走行、方向転換、ステップ、ジャンプ
- 歩行動作の生成を目的とした手法が開発されている



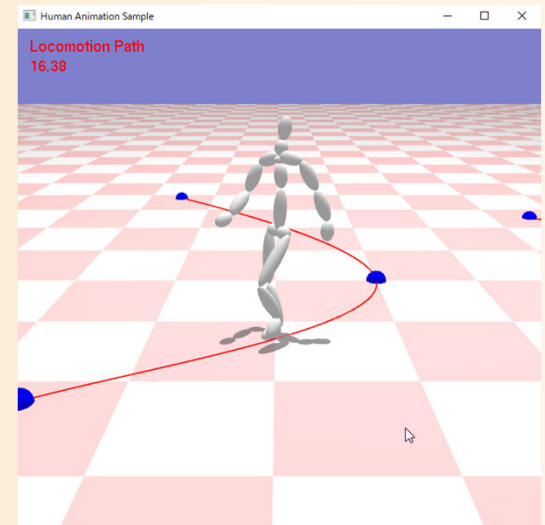
歩行動作生成の手法

- 動作補間による歩行動作生成
 - 速度や方向などの特徴パラメタによる動作補間
- 動作変形による歩行動作生成
 - 軌道や地面の高さに合わせた動作変形
- 動力学シミュレーションによる手法
 - 手作業で設計されたコントローラの利用
 - 機械学習によるコントローラの獲得
- 深層学習による動作生成



デモプログラム

- 軌道に沿った歩行動作の生成
 - 1サイクル分の歩行動作の軌道を変形して、入力軌道に当てはめる
 - 歩行軌道の編集
 - 制御点をマウスでドラッグすることで操作
 - B-Spline曲線による軌道生成
 - 歩行動作の生成



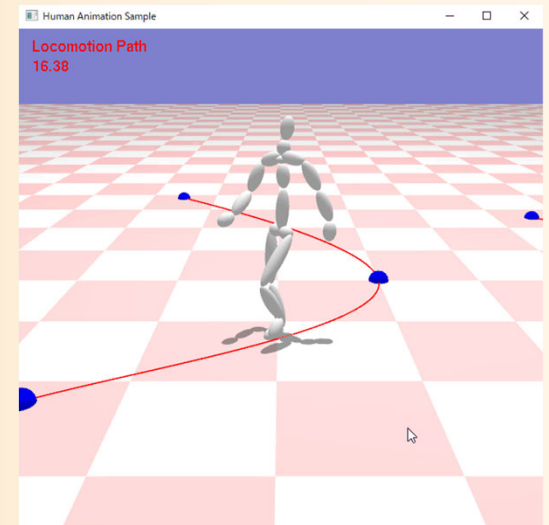


軌道に沿った歩行動作
Locomotion Along Path



軌道に沿った歩行動作の生成(1)

- デモプログラムでの、歩行動作の生成方法
 - 1サイクル分の歩行動作の軌道を変形して、入力軌道に当てはめる
 - 軌道上の各点における、歩行動作の状態を計算
 - 時刻、位置・向き、歩行動作のステップ数、正規化時間
 - 再生時刻に対応する点から、歩行動作の姿勢を生成
 - 曲率が高い箇所では、動作が不自然になる可能性がある



軌道に沿った歩行動作の生成(2)

- 歩行軌道上の点での状態を表すデータ構造

```
// 軌道上の各点での歩行動作の状態を表す構造体
struct LocomoPoint
{
    // 歩行開始からの経過時間
    float    time;

    // 歩行動作の繰り返し回数
    int      step_count;

    // 歩行動作の正規化時間(0.0~1.0)
    float    step_norm_time;

    // 位置・向き(基準点の位置・向き)
    Point3f  pos;
    float    ori;
}

// 歩行軌道上の各点の状態の配列
vector< LocomoPoint > locomotion_points;
```



軌道に沿った歩行動作の生成(3)

• 歩行動作の初期化

- 入力軌道を連続的な点の集まりで表す
- 各点での歩行動作の状態を求める
 - 1サイクル分の歩行動作データを読み込む
 - 1サイクル分の歩行動作データの移動距離や時間から、各点の時刻や歩行動作の繰り返し回数・正規化時刻を計算する

• 歩行動作の再生

- 現在時刻の点での歩行動作の状態を、前後の点の状態を補間して求める
- 歩行動作から取得した姿勢を、現在時刻の点の位置・向きに当てはめる





深層学習を用いた動作生成

深層学習を用いた動作生成

- 機械学習の動作データ処理への利用
 - 動作認識(分類)には、古くから利用されている
 - 動作データは複雑であるため、従来の機械学習手法では、動作生成への応用は難しかった
 - 最近では、深層学習(ニューラルネットワーク)を動作生成に応用する技術も開発されている
 - 多数の動作データを学習することで、何らかの入力にもとづいて、動作を生成
 - 例: 移動方向・速度に応じた歩行動作の生成など
 - 実用レベルでの利用は、まだ少ない



動作データの形式

- 機械学習を適用する場合、動作データを別の形式に変換して扱うこともある

- 信号データ

- 各関節の角度(1次元)の時間変化

- グラフデータ

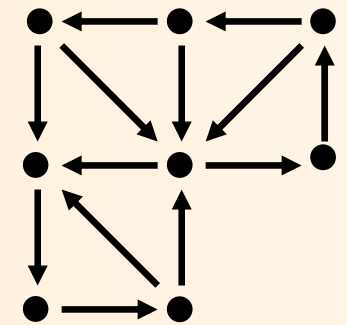
- 各ノードが姿勢や短い動作を表す

- 姿勢状態空間上の点や軌道データ

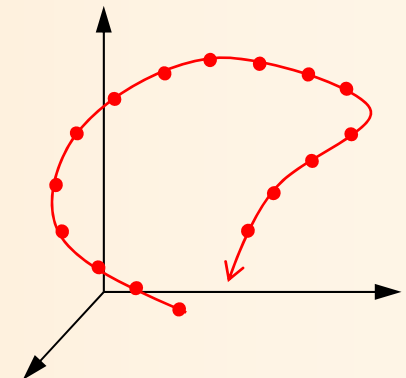
- 高次元空間、低次元空間への圧縮

- 画像データ

- 一定時間の動作を画像に変換(後述)



動作グラフ構造



動作状態空間



深層学習の適用時の問題点

• 動作データへの深層学習の適用例

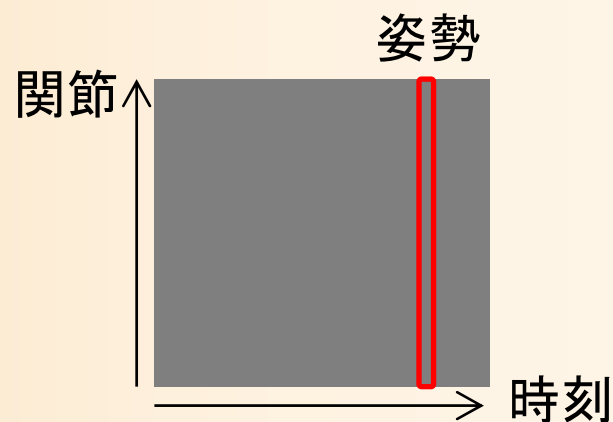
– 一定時間の動作データを画像として表現する

- 全関節の回転情報を一列にまとめる

 - 回転の代わりに位置や回転・並進速度を使うこともできる

- AutoEncoder や GAN を利用した動作生成

- いずれにしても、生成後の画像では姿勢・動作としての制約が保証されないため、姿勢・動作変形の後処理が必要になる

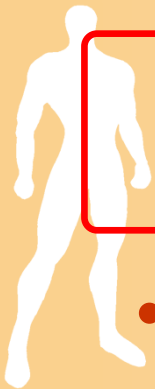


– 各時刻の姿勢を入出力として、時系列データを扱うネットワーク(RNNなど)を利用する方法もある



今回の内容

- キャラクタアニメーションの課題
- 動作生成・制御
 - 動作状態機械
 - 動力学を考慮した動作生成
 - 歩行動作生成
 - 深層学習による動作生成
 - 自律動作制御
 - 身体制御、動作制御、行動制御
- 講義のまとめ





自律動作制御

自律動作制御

- 自律動作制御

- 与えられた指示や周囲の状況に応じて、適切な動作を生成
- 例：コンピュータゲームにおける自律動作制御



Non-Player Character

完全に自律的に動作を行う必要がある

Player Character

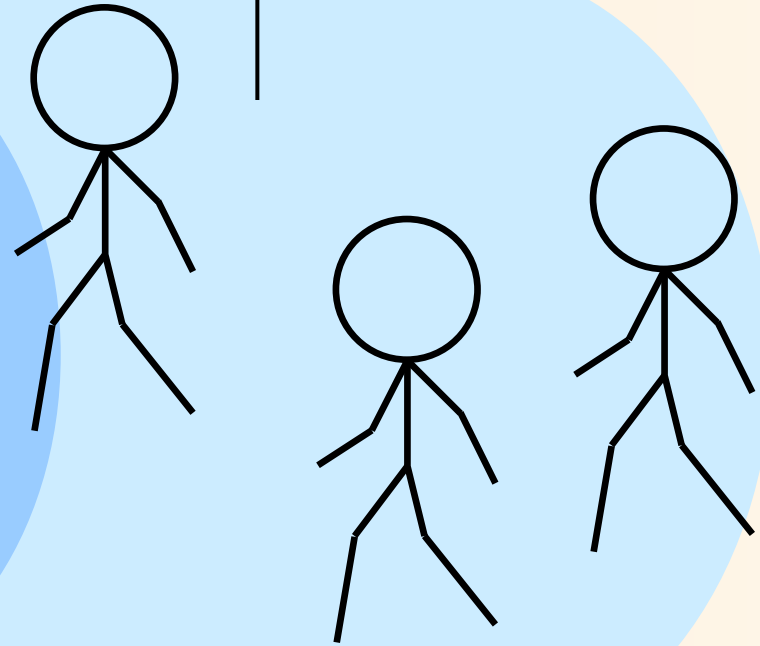
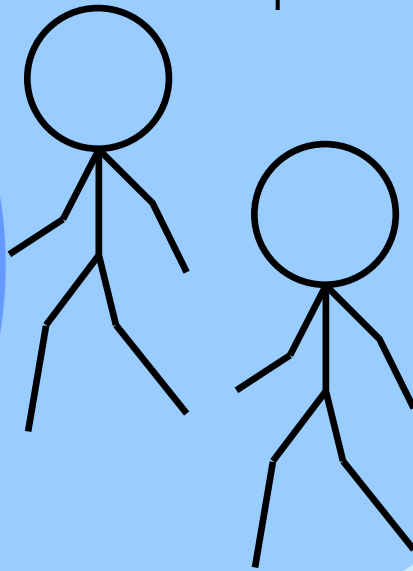
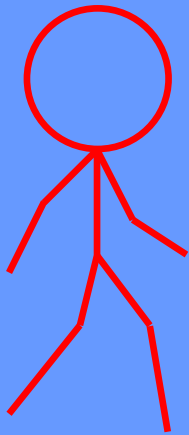
半自律的に動作を行う必要がある
(ex. プレイヤーの抽象的な操作に応じて具体的な動作を実現)

自律動作制御のレベル

身体制御

動作制御

行動制御



自律動作制御のレベル

身体制御

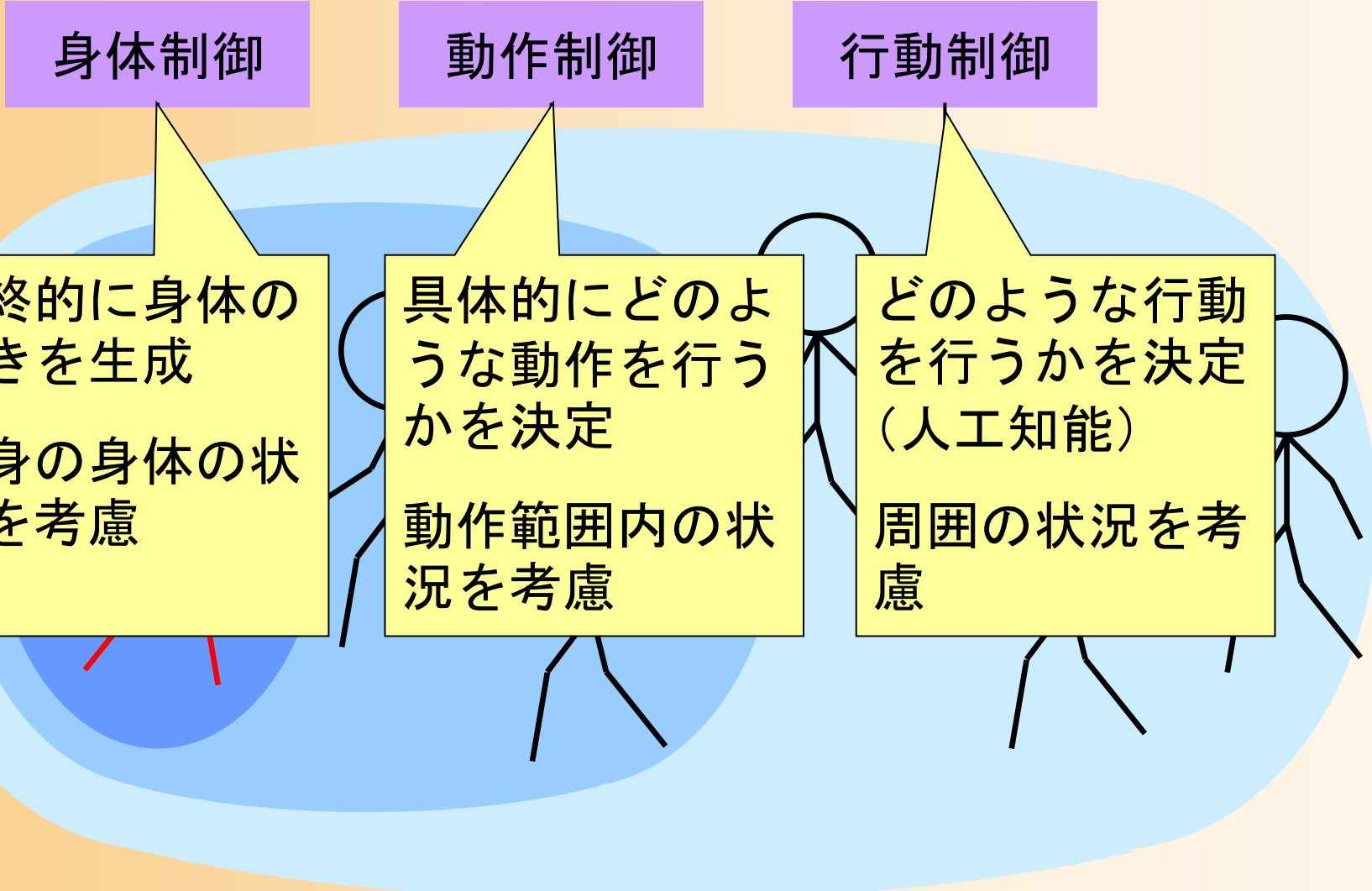
最終的に身体の動きを生成
自身の身体の状態を考慮

動作制御

具体的にどのような動作を行うかを決定
動作範囲内の状態を考慮

行動制御

どのような行動を行うかを決定
(人工知能)
周囲の状態を考慮



身体制御の技術

身体制御

最終的に身体の動きを生成

自身の身体の状態を考慮

動作制御

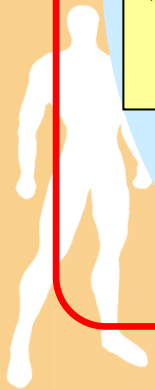
具体的にどのような動作を行うかを決定

動作範囲内の状態を考慮

行動制御

どのような行動を行うかを決定
(人工知能)

周囲の状態を考慮



身体制御の課題

- 衝突や外力などの力学的な影響に応じた身体制御(動作生成)
 - 動力学シミュレーションを使う方法では、受動的な動作は生成できるが、能動的な動作は生成できない



身体制御の技術

- モデルベース手法 vs データベース手法

- モデルベース手法

- 何らかのモデル(アルゴリズム)に従って動作を生成
- 個人ごとの動作の違いの実現などは困難

- データベース手法

- 大量のデータを用意しておくことで、問題を解決
- 準備が必要、適切なデータの検索・変形は困難

- ハイブリッド手法

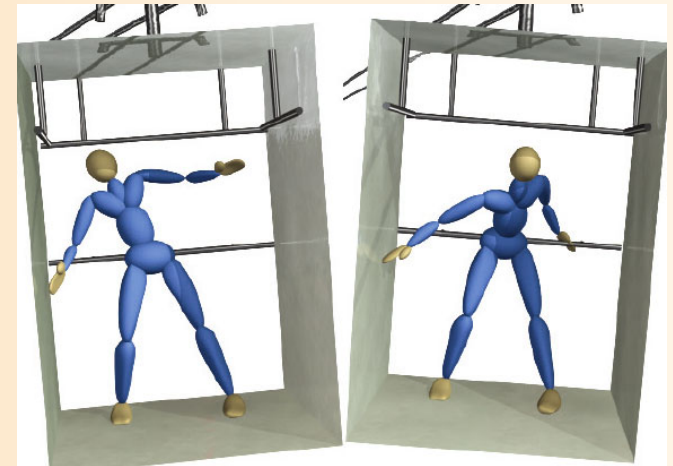
- 両方をうまく組み合わせる手法
- どのように組み合わせるかは、難しい



関連研究(モデルベース手法)

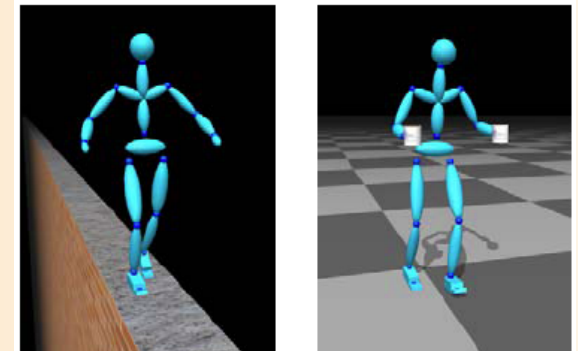
- 関節角度制御 [Jain 2009]

- 力学的要素を考慮して、最適な角度(+速度、加速度)を決定
- 最適化問題に帰着



- 歩行制御モデル [Wang 2009]

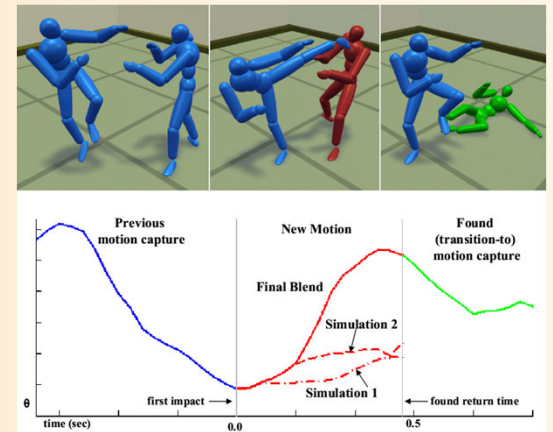
- 動力学シミュレーションによる歩行動作
- 安定性を上げるための工夫



関連研究(ハイブリッド手法)

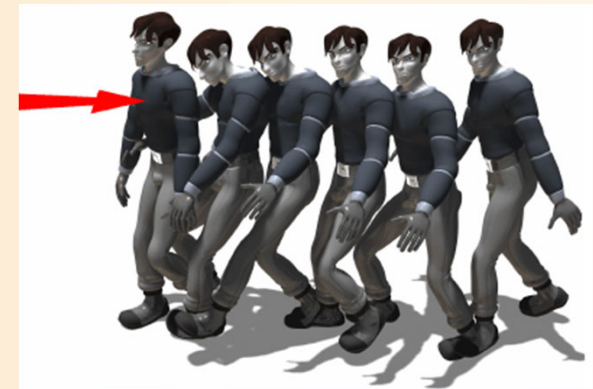
- シミュレーションと動作データの利用 [Zordan 05]

- 衝撃後の動作を動力学シミュレーションで計算
- 後にうまくつながるような転倒動作をデータベースから検索



- 動作データの選択と変形 [Arikan 05]

- 衝撃・姿勢に応じたリアクション動作データを検索
- 各部位に一定の速度を加えて逆運動学により姿勢を変形



関連技術(ハイブリッド手法?)

- Endorphin, Euphoria
(Natural Motion 社)

- タイムライン上でビヘイビアを指定すると自動的に動作生成



Natural Motion Ltd

- さまざまなビヘイビアの制御モデルが用意されており、ビヘイビアの種類やタイミングを指定することで、自動的に動作が生成される(Endorphin)
- ゲーム用のミドルウェアも存在(Euphoria)
- 詳細な内部技術は公開されていない
- モデルベースなので、動作を細かく変更したり、キャラクターの個性を出したりすることは難しい?



ロボット工学との関連

- ロボット制御とはやや目的や手段が異なる
 - ヒューマノイドロボット制御
 - 力学的に正確な制御が必要
 - 現在では、ごく単純な動作しか実現できていない
 - 人間らしい動作の再現はあまり考えられていない
 - 人間とはモータの機構も異なる
 - アニメーション
 - 力学的には厳密でなくても、人間らしい動きを実現
 - 動力学などの基礎的な理論はかなり共通
- 将来的にはヒューマノイドロボットの技術がゲームにも応用可能になる？



動作制御

身体制御

最終的に身体の動きを生成

自身の身体の状態を考慮

動作制御

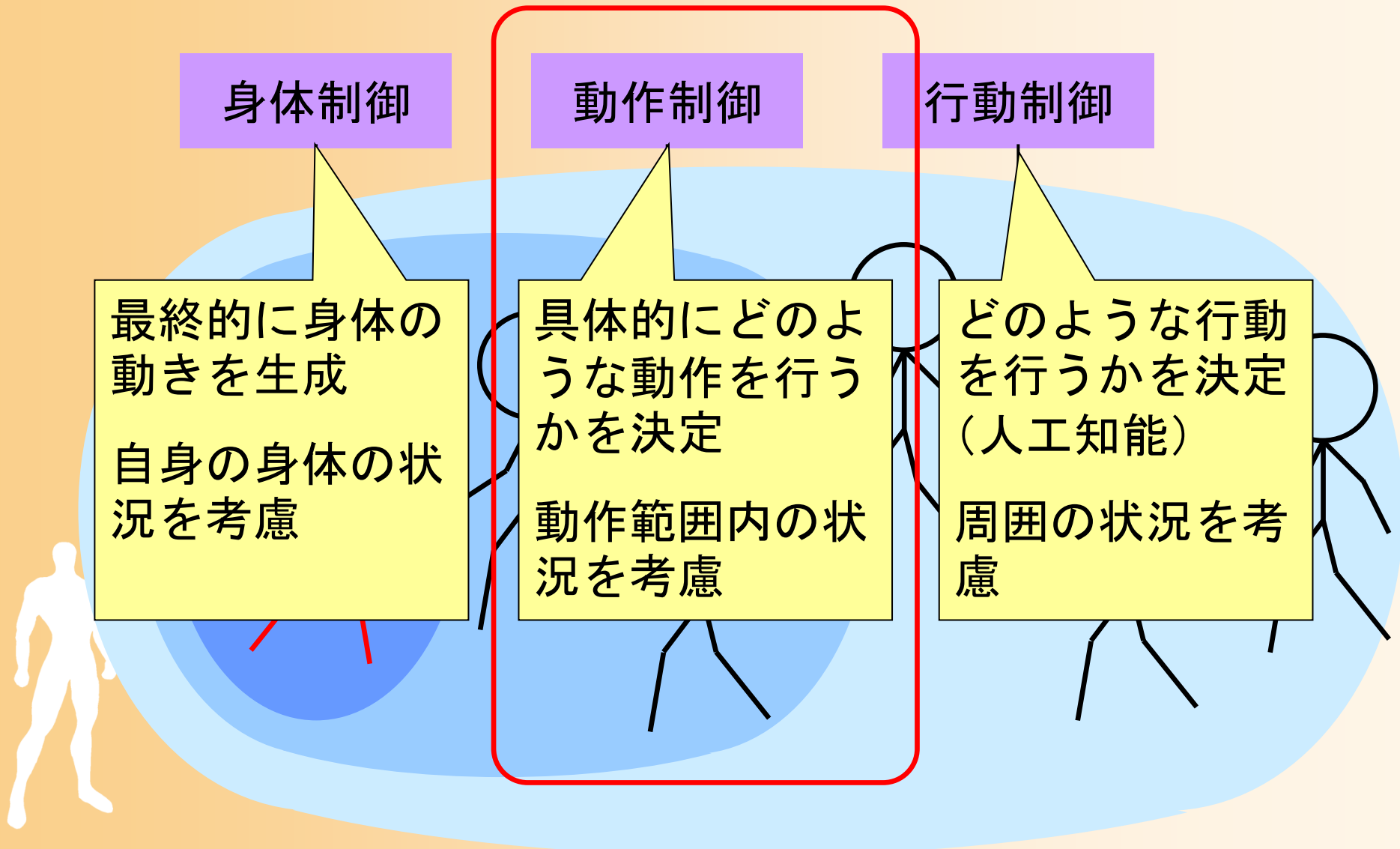
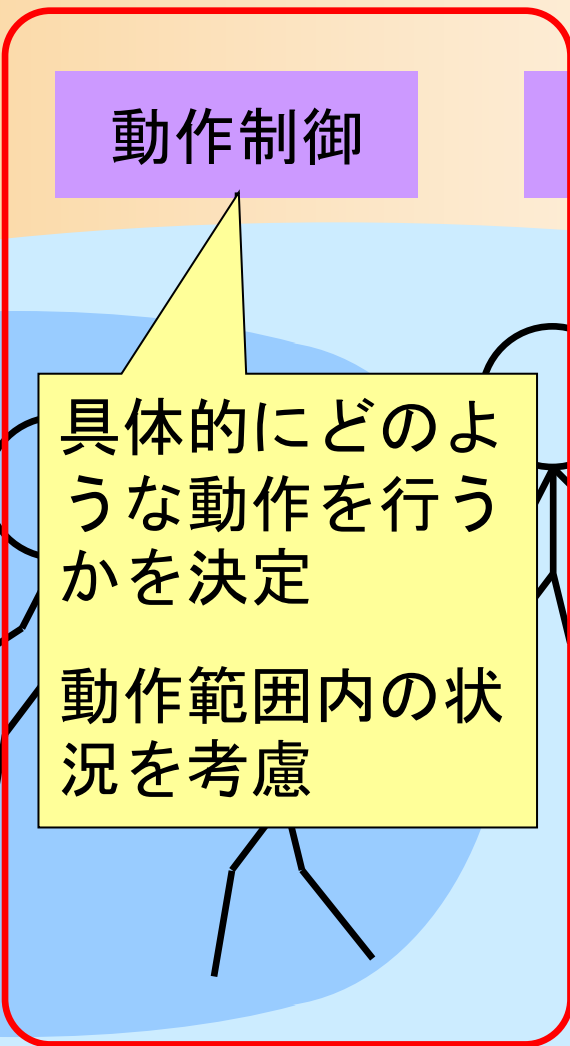
具体的にどのような動作を行うかを決定

動作範囲内の状態を考慮

行動制御

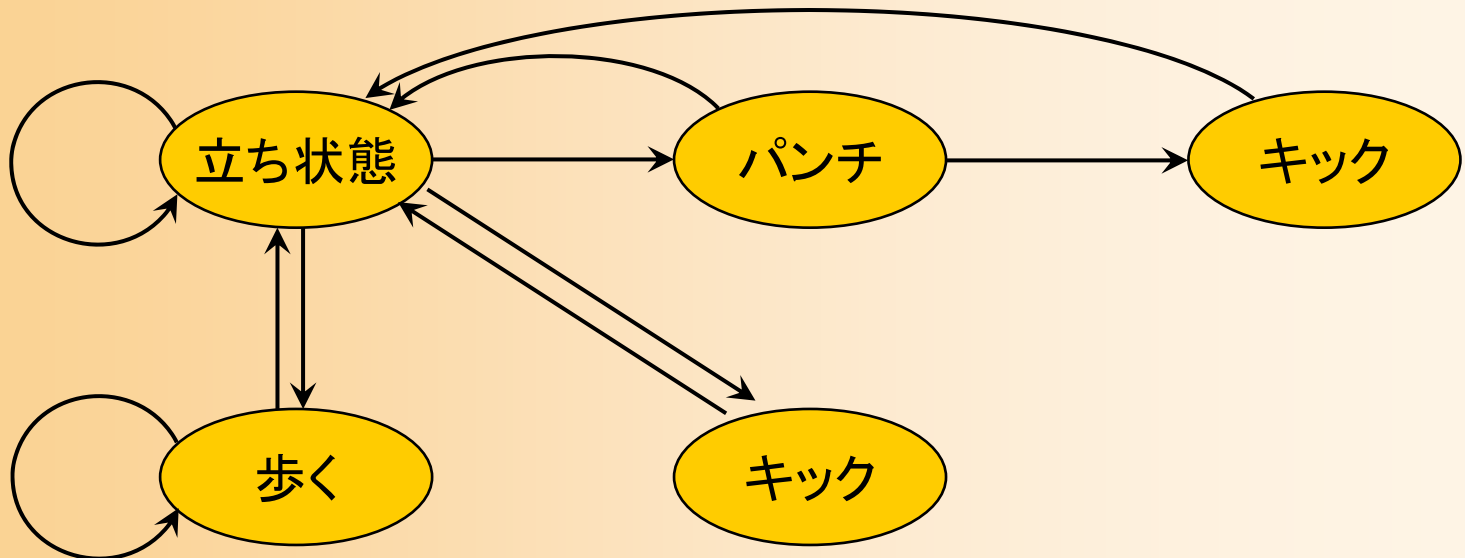
どのような行動を行うかを決定
(人工知能)

周囲の状態を考慮



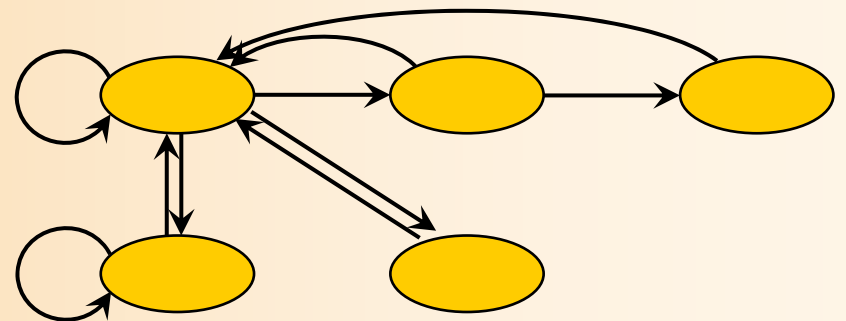
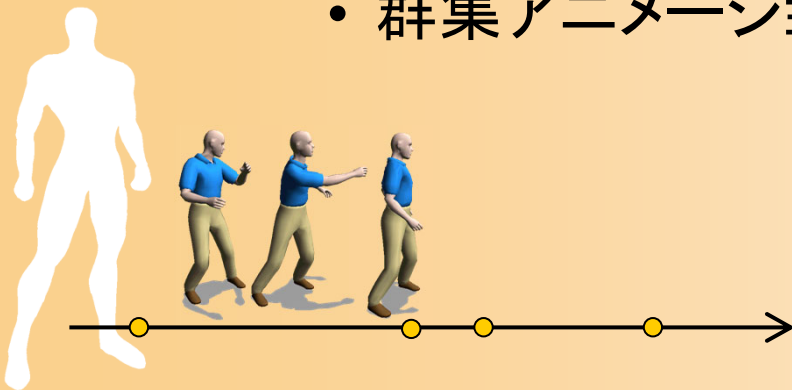
動作制御の課題

- 動作状態機械を用いた動作制御
 - 限られた種類の動作しか行うことができない
 - 動作状態機械の作成に手間がかかる
 - 状況に応じた適切な動作の選択が必要



動作状態機械の自動作成

- 動作データを解析して、動作状態機械的なグラフ構造を自動的に作成する技術もある
→ モーショングラフ
 - 自動的に生成されたグラフ構造は整理されていないため、ゲームなどへの利用は難しい？
 - キャラクタを自律的に動かすような用途に有効
 - 群集アニメーションなど

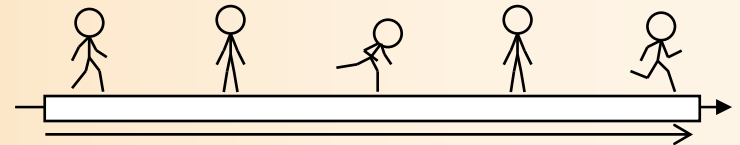


モーショングラフ

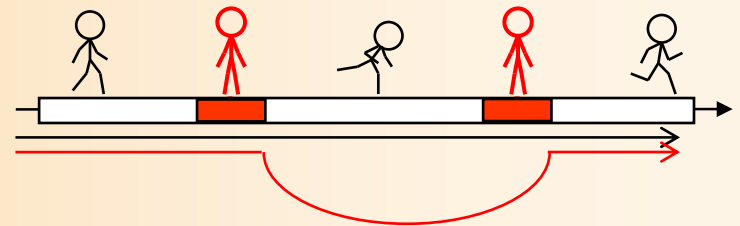
- **モーショングラフ [Kovar 02]**

- モーションキャプチャデータを、有向グラフ構造に変換できる
- ノードを順に遷移することで、連続的な動作を生成できる
 - 遷移のルールが重要

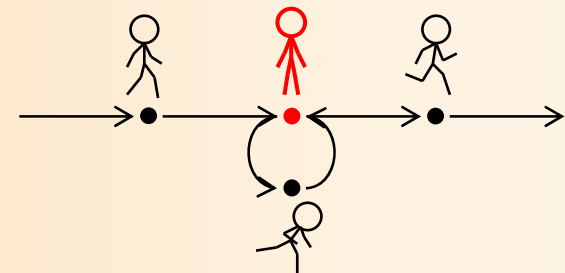
(a) 動作データは単に再生しかできない。



(b) 類似部分があれば、切り替え可能



(c) 類似部分に注目しグラフ構造に変換

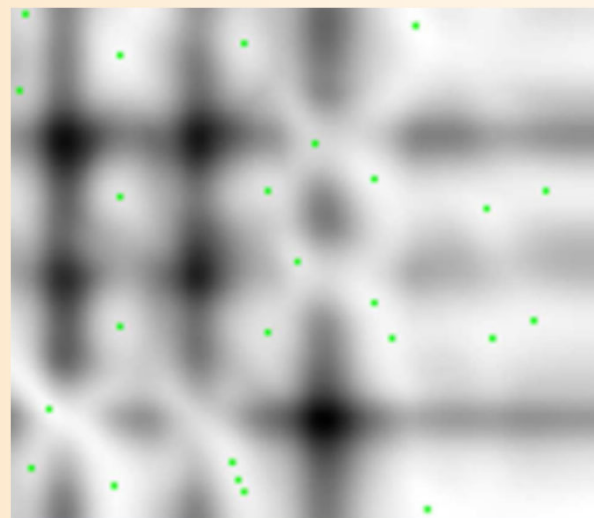


モーショングラフの作成方法

- 2つの動作中の各フレーム同士の姿勢間の距離を計算(マッチウェブ)

- 姿勢同士の距離の評価方法も重要

- 例: 主要部位同士の距離の和、ワンスキンモデルの頂点同士の距離の和、など



Kovar 02

- マッチウェブ中の極小点をモーショングラフのノードとし、ノード間の動作をエッジとする



姿勢間の距離の計算(1)(復習)

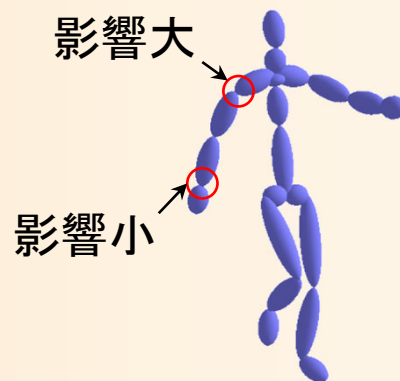
- 姿勢表現にもとづいて評価することは難しい
 - 全関節角度の差の平均値による評価

$$D = \frac{1}{n} \sum_i |\theta_i^1 - \theta_i^2|$$

n 関節数
 θ_i^1 θ_i^2 各姿勢の i 番目の関節の回転角度

- 関節によって、姿勢の見た目に与える影響には違いがあるため、このような計算方法は不適切

- 関節の重み付けを行うことも難しい



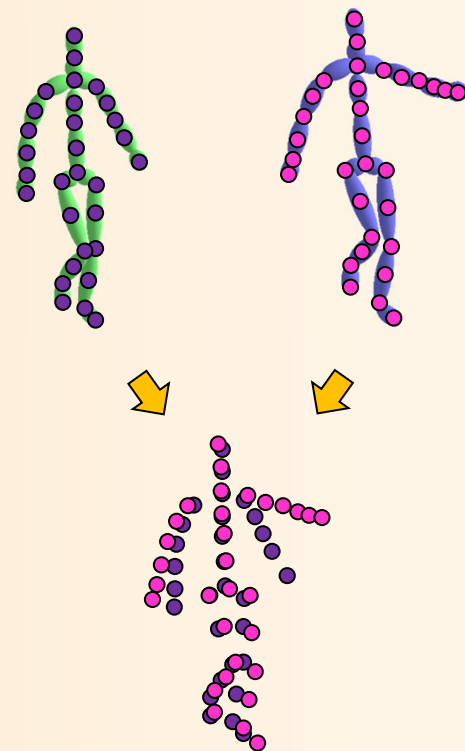
- 姿勢を点群で表現して、その点同士の平均距離によって評価する方法が用いられる



姿勢間の距離の計算(2) (復習)

1. 姿勢を点群で表現

- 関節・体節位置
- 形状変形モデルの頂点、など



2. 位置・方向が一致するように一方の点群を移動・回転

- 2次元平面上での正規分布を求めて、分散が一致するように移動・回転

3. 点同士の距離の平均を計算

$$D = \frac{1}{m} \sum_i |\mathbf{p}_i^1 - \mathbf{p}_i^2|$$

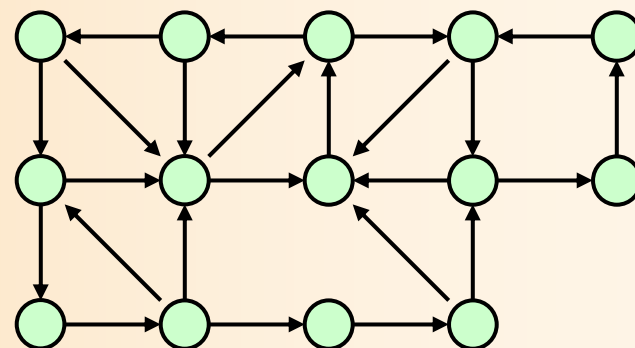
m 点の数
 \mathbf{p}_i^1 \mathbf{p}_i^2 各姿勢の i 番目の点の座標



モーショングラフによる動作生成

- データ構造

- ノード・・・姿勢
- エッジ・・・動作
 - 動作状態機械とは逆



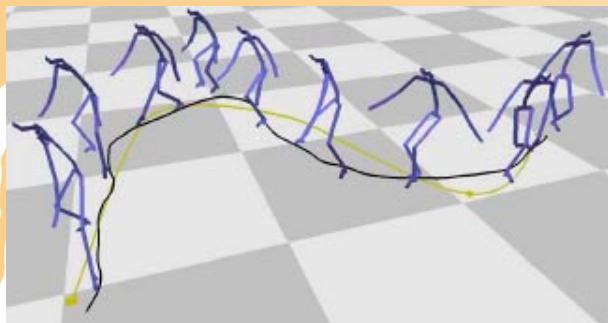
- 動作生成

- ノードを辿りながらエッジの動作を再生していくことで、動作を生成
 - 実際には、ノードの前後で、動作ブレンディングや、足を地面に固定するための逆運動学計算が必要
- ノード遷移のためのルールが重要

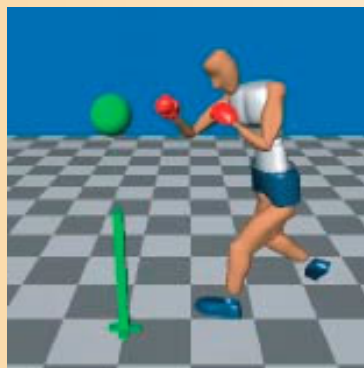


関連研究

- モーショングラフにおけるノード選択条件
 - 歩行軌道による条件 [Kovar 02]
 - 手先位置の条件 [Lee 05]
 - 動作の種類別の条件 [Arikan 04]
 - 音楽に合わせた動作



[Kovar 02]



[Lee 04]



[Arikan 04]

モーショングラフの利点

- 動作データから自動的に構築できる
- 連続的な動作が無限に生成できる
 - 毎回異なる動作が生成される
 - 基本的には同じ動作の繰り返しだが、ノード数が十分にあれば、十分に自然に見える
- デメリット
 - 毎回異なる動作が生成される
 - ゲームによっては必ずしも良いとは限らない？
 - 適切な遷移ルールが必要(計算速度の問題)



行動制御

身体制御

最終的に身体の動きを生成

自身の身体の状態を考慮

動作制御

具体的にどのような動作を行うかを決定

動作範囲内の状態を考慮

行動制御

どのような行動を行うかを決定
(人工知能)

周囲の状態を考慮



行動制御の課題

- 周囲の状況に応じた行動の決定が必要
- 用途に応じた、行動制御のモデルが必要



群集モデル

- 多数の人間の動きをシミュレートする技術
 - 映画やコンピュータゲームでの群衆シーンで利用
 - 個々のキャラクターの動作を作成するのは不可能
 - 一定のルールに従って各キャラクターを制御
 - 単純なルールでも、キャラクター同士が相互作用し合うことで、ある程度リアルな動きが生成される



Lord of the Rings, 2002



Chronicles of NARNIA, 2005



NINETY-NINE NIGHTS, 2005

群集モデルの実現方法

- プログラムによる動作ルールの記述
 - if (条件) then (動作) の形式の動作ルールを多数、ゲームなどのプログラム内に直接記述
- GUI環境による高度な動作ルールの記述
 - 群衆シミュレーションソフトMassive が有名
 - 多数の映画で使用されている
 - 状態機械＋ファジィルールによる条件を、GUI画面上で記述
 - かなりの知識や労力が必要



Massive



群衆シミュレーションの手法

- エージェントモデル

- 各キャラクタ(エージェント)を点とみなして、一定のルールに従って点の移動を計算
 - social force model, flocking model など

- セルオートマトン

- 空間を細かいグリッド(セル)に分けて、エージェントが存在するセルを、隣接するセルに移動させる

- 流体モデル

- 流体力学にもとづき、密集した群衆の移動を計算

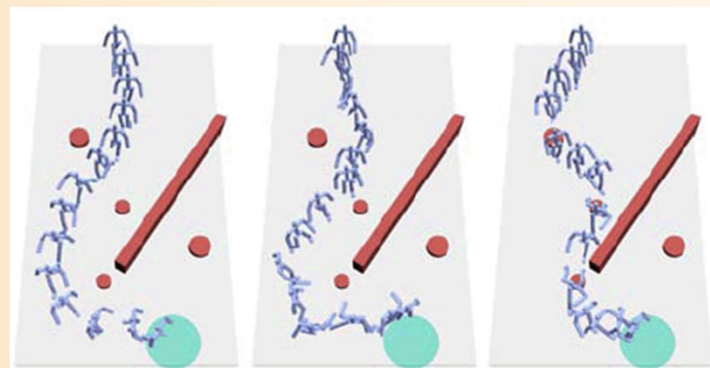


関連研究

- 行動ルールの自動学習

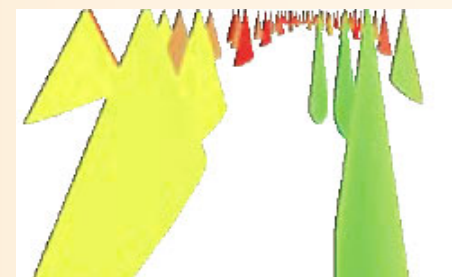
- 逆強化学習 [Lee 2010]

- 動作計画のための
評価基準を自動学習



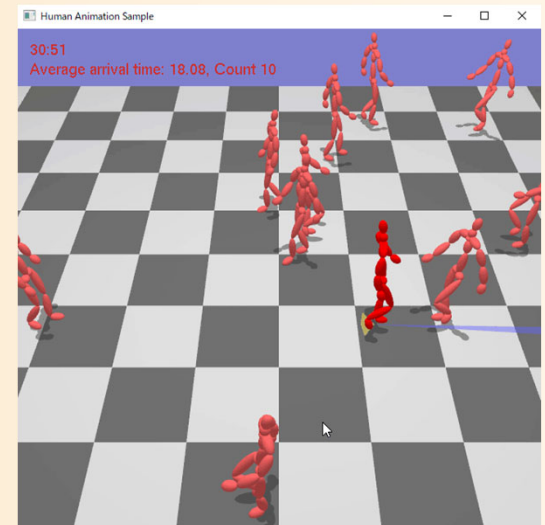
- 高度なルールベースの手法

- 視界にもとづく動作
決定 [Ondrej 2010]



デモプログラム

- エージェントモデルによる群衆シミュレーション
 - Social Force Model [Helbing 1995]
 - エージェントに働く複数の力を計算して加算
 - 目標位置・目標速度を満たすための推進力
 - 周囲のエージェントや壁・障害物と離れる反発力
 - エージェントに働く力から、加速度を計算し、速度・位置を更新(粒子シミュレーション)
 - エージェントを表す点の移動に歩行動作を当てはめて再生



18:05

Average arrival time: 15.37, Count 1

群衆シミュレーション Crowd Simulation



自律動作制御のまとめ

身体制御

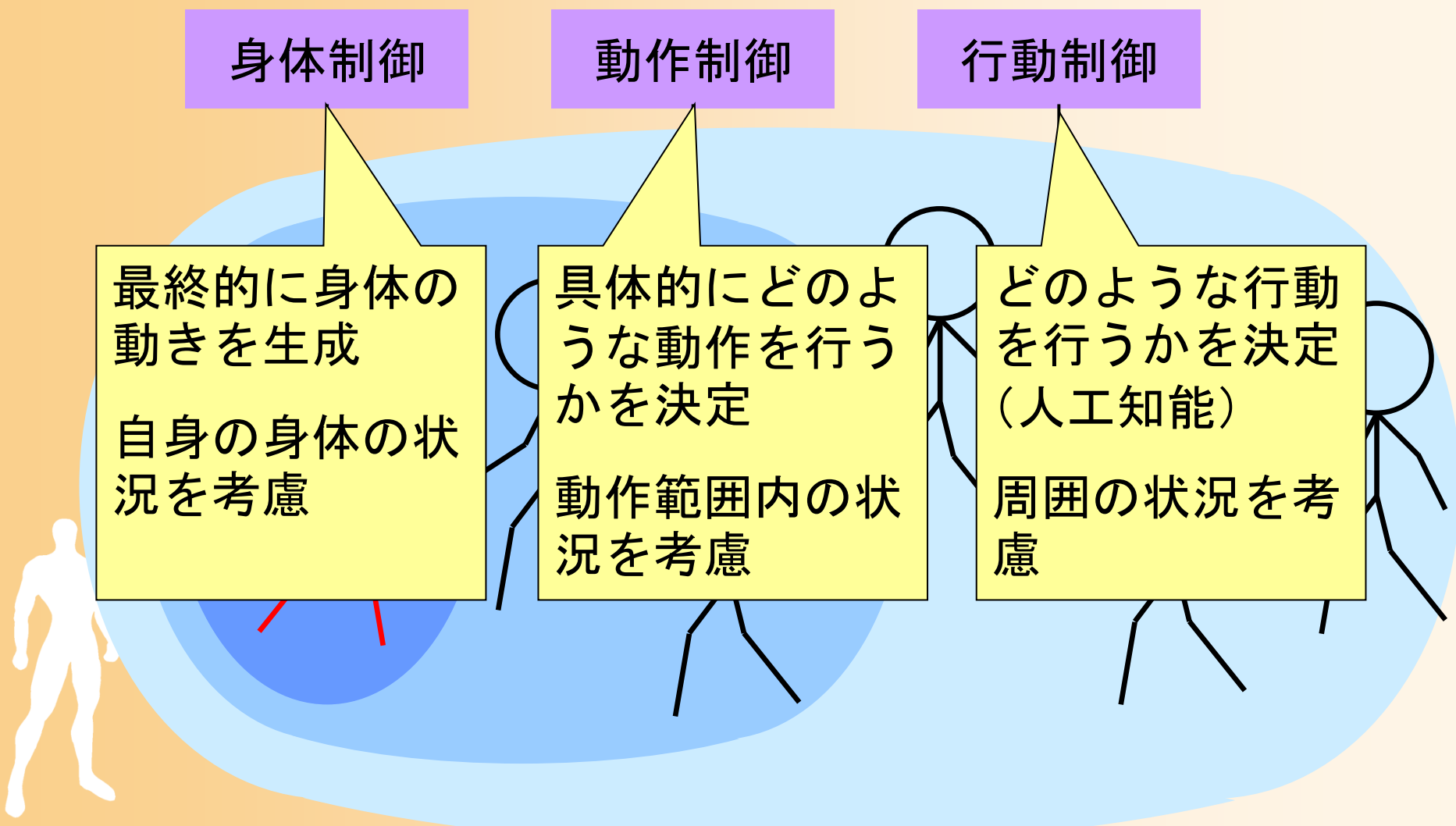
最終的に身体の動きを生成
自身の身体の状態を考慮

動作制御

具体的にどのような動作を行うかを決定
動作範囲内の状態を考慮

行動制御

どのような行動を行うかを決定
(人工知能)
周囲の状態を考慮



まとめ

- 動作生成・制御
 - 動作状態機械
 - 動力学を考慮した動作生成
 - 歩行動作生成
 - 深層学習による動作生成
 - 自律動作制御
 - 身体制御、動作制御、行動制御

- 講義のまとめ



全体のまとめ

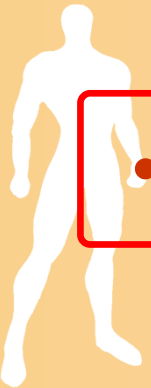
- 人体モデル(骨格・姿勢・動作)の表現
- 人体モデル・動作データの作成方法
- サンプルプログラム
- 順運動学、人体形状変形モデル
- 姿勢補間、キーフレーム動作再生、動作補間
- 動作接続・遷移、動作変形
- 逆運動学、モーションキャプチャ
- 動作生成・制御



今回の内容

- 動作生成・制御
 - 動作状態機械
 - 動力学を考慮した動作生成
 - 歩行動作生成
 - 深層学習による動作生成
 - 自律動作制御
 - 身体制御、動作制御、行動制御

- 講義のまとめ





講義のまとめ

本科目の達成目標

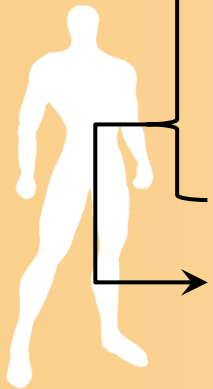
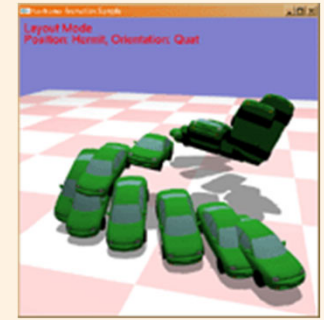
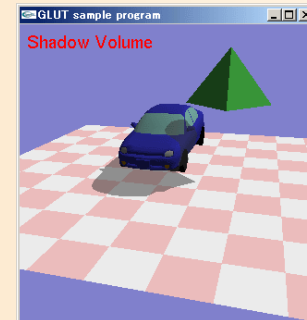
- 最近の3次元CGで使われている応用技術について学ぶ
- 学習した技術を実際に応用して、3次元CGを使ったプログラムを作成できるようになる
- コンピュータグラフィックス技術を用いるソフトウェアを作成するときに必要なとなる、実用的な技術を中心に学習する



本科目の内容

- 本科目で学ぶ応用技術
 - 視点操作
 - 幾何形状データの読み込み
 - 影の表現(高度な描画技術)
 - キーフレームアニメーション
 - 物理シミュレーション
 - 衝突判定、ピッキング
 - キャラクタアニメーション

特に(キャラクタ)アニメーション
関連の技術を重点的に扱う



コースとモジュール

- 本科目は「グラフィックスと応用」モジュールに属する
 - 本モジュールに属する他の講義
 - マルチメディア表現／工学特論(乃万先生)
 - ヒューマンインタフェース(大橋先生)
 - 仮想空間論(碓崎先生)
 - コンピュータビジョン特論II(岡部先生)
 - (5科目中3科目の単位を取得すれば、モジュール修了)
- 「グラフィックスと応用」モジュールは、「メディア処理」コースに属する
 - 他のモジュールも修了すれば、コース修了認定



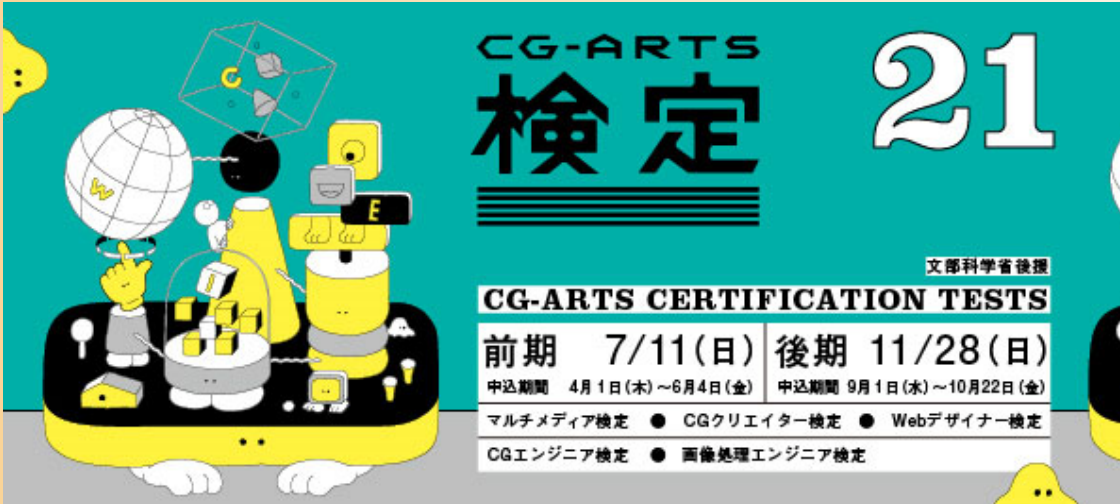
成績評価

- プログラミング演習課題(80%)
 - 演習課題のプログラムを作成し、レポートを提出
 - レポート提出後に演習問題(ミニテスト)を実施
 - 5回程度のレポート課題を予定
- 演習問題(20%)
 - 毎回の授業に出席して、講義内容を理解する
 - 授業中に演習問題(ミニテスト)を実施
 - 上記以上の詳細や、個別の成績評価に関する質問には回答しない



参考:CGエンジニア検定

- CGエンジニア検定(CG-ARTS協会)
 - ベーシック/エキスパート
 - 年2回実施、福岡でも受験可能
 - 学部のコンピュータグラフィックス科目の内容を修得できていれば合格可能



CG-ARTS CERTIFICATION TESTS	
前期 7/11(日)	後期 11/28(日)
申込期間 4月1日(木)~6月4日(金)	申込期間 9月1日(木)~10月22日(金)
マルチメディア検定 ● CGクリエイター検定 ● Webデザイナー検定	
CGエンジニア検定 ● 画像処理エンジニア検定	

授業アンケートへの回答

- 大学院 授業アンケート
 - LiveCampusにログインして回答
 - 回答期間は掲示を参照
- 成績評価には一切影響しません



まとめ

- 動作生成・制御
 - 動作状態機械
 - 動力学を考慮した動作生成
 - 歩行動作生成
 - 深層学習による動作生成
 - 自律動作制御
 - 身体制御、動作制御、行動制御

• 講義のまとめ

