CEDEC 2010
CESA Developers Conference
×
サカつく.com
www.sakatuku.com

「サカつく」のサッカー試合AIシステム

株式会社セガ
第一CS研究開発部
安藤 毅

CEDEC 2010
CESA Developers Conference
×
サカつく.com
www.sakatuku.com

はじめに

「サカつく」とは？
「サカつく」のサッカー試合とは
本講演で得られる知見

「サカつく」とは？

- 公式HP「サカつく.com」より抜粋
  http://www.sakatsuku.com/

「サカつく」とは？

本講演では、これらの要素には触れません。

「サカつく」のサッカー試合とは

サッカー試合の
リアルタイムシミュレータ

- チーム育成の成果を確認する場
- 操作の優劣で結果が大きく左右されてはならない

- ユーザーは監督として間接的な指示を出せるのみ
- 選手個々のプレーには直接関与できない

「サカつく」のサッカー試合とは

- 「サカつく」試合に求められる要件
  - 試合シミュレーターとしての“納得感”
    - 強いチームの勝率が高い
    - サッカーらしいスコア
  - 映像としての面白さ
    - サッカー中継のようなハラハラ・ドキドキ感を提供
    - CGならではの表現（カメラ、エフェクト）
  - 処理負荷の軽さ
    - 「結果をみる」モードへの対応
    - 「試合を見る」モードと結果が一致するのが望ましい

これら要件を満たした試合システムを
どのように設計するか？

本講演で得られる知見

- リアルタイムシミュレーションに適したAIシステム設計の一手法

  「サカつく」シリーズの試合システムの変遷を通じて、生じた課題、解決手法を紐解きます

- メモリや処理速度などのリソース制約下における設計の一手法

  「サカつくDS」というタイトル制作において気を遣ったポイント、設計手法を解説します

本講演では、探索アルゴリズムや学習、協調動作等の高度なAI分野にはいっさい触れません。ご了承ください。

「サカつく」試合システムの変遷

方法１：完全オーサリング方式
方法２：完全リアルタイム方式
方法３：折衷方式

方法１：完全オーサリング方式

- 採用シリーズ
  - 「J.LEAGUE プロサッカークラブをつくろう!」(1996,セガサターン)～「 J.LEAGUE プロサッカークラブをつくろう!’ 04」(2004,PS2) の各CSタイトル
  - 「J.LEAGUE プロサッカークラブをつくろう! アドバンス」(2002,GBA)
  - 「J.LEAGUE プロサッカークラブをつくろう! 6 Pride of J」(2009,PSP)
- 概要
  - オーサリング済みのシーンを試合場面ごとにたくさん用意し、結果に応じて選択表示

方法１：完全オーサリング方式

- 利点
  - クオリティの高い試合シーンが制作できる
    - 人間らしい一連の動き、複数人が絡むモーション表現
    - カメラワークなどの演出を狙って作れる　光プレイ!
  - AI処理が見栄えに影響を与えない
    - ➢処理負荷の軽いAIを「結果を見る」モードと共有できる
- 問題点
  - シーンのバリエーション増で対応

同じシーンばかり再生される

展開が読めてしまう

制作工数の肥大化

方法２：完全リアルタイム方式

- 採用シリーズ
  - 「プロサッカークラブをつくろう! ヨーロッパチャンピオンシップ」(2006,PS2)
  - 「 J.LEAGUE プロサッカークラブをつくろう! 5」(2007,PS2)
- 概要
  - アクションゲームと同じアプローチ
  - 個々の選手がモーションの状態管理、動作判断をリアルタイムに行う

方法２：完全リアルタイム方式

- 利点
  - 自由度の高さ
    - 戦術設定・AIしだいで、局面は無限に存在
    - 先の読めないサッカーの醍醐味を演出
- 問題点

  - 試合結果の間接的なコントロール
    - 結果のバランス調整がきわめて困難
  - 処理速度の向上に限界
    - 「結果をみる」モードの実現に苦慮

自由すぎた…

方法３：折衷方式

- 採用シリーズ
  - 「サカつくDS　タッチandダイレクト」(2008,NintendoDS)
  - 「サカつくDS　ワールドチャレンジ2010」(2010, NintendoDS)
- 設計思想
  - 過去の２方式の「良いとこどり」を模索
    - オーサリング方式の…
      - シーンの見栄え
      - 見栄えに影響しないAI処理
    - リアルタイム方式の…
      - 自由度の高さ

さて、どうする？

本章のまとめ

- 「完全オーサリング方式」「完全リアルタイム方式」は、それぞれに固有の利点・問題点がある
- 新システムを設計するにあたり、それぞれの利点を引き継ぎ、かつ問題点を引き継がない「折衷方式」を目指した

以後、この折衷方式について詳しく解説していきます

システム概要

「サカつくDS」固有の要件
設計のコンセプト
プレイセットとは？
局面計算の流れ
局面計算の結果管理
グラフィック再生

「サカつくDS」固有の要件

- **ユーザー介入**を重視した試合システム

選手の
フォーメーションを
ダイナミックに
変えられる！

チームスキルを
使用して
試合を有利に！

「サカつく」シリーズの中ではアクション性が高い。

設計のコンセプト

- “ほぼ”リアルタイム性
  - ユーザー操作をこまめに結果に反映
  - フレーム単位のレスポンスは求めない
- ハードウェア処理能力による制約
  - そもそも全員を賢く制御するのは困難

どうせなら
球際だけでも
美しく

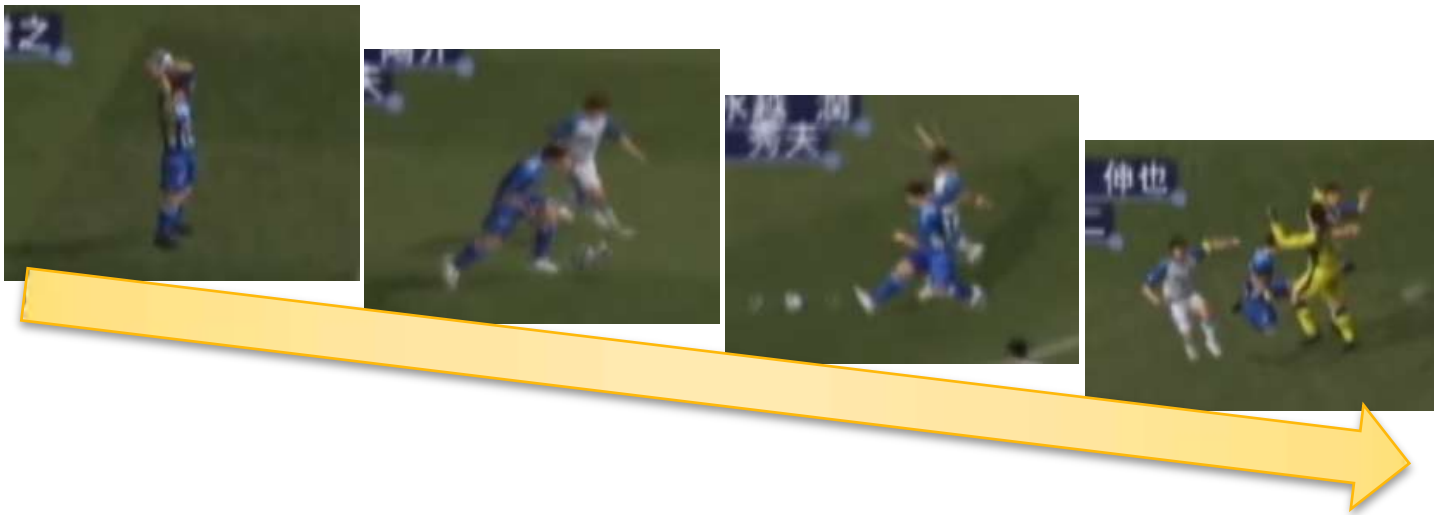
設計のコンセプト

- “球際だけオーサリング”方式

  - 「ボール保持者＋最小限の選手」だけの小さなシーン
  - ほかの選手は注目されないので簡単な状態管理にとどめる

設計のコンセプト

- “球際だけオーサリング”方式
  - １プレーごとの短いシーン
    - シーンの再利用を容易に
    - 最低限のレスポンスに対応
    - 連続再生でプレーが繋がって見えるエ夫は必要



設計のコンセプト

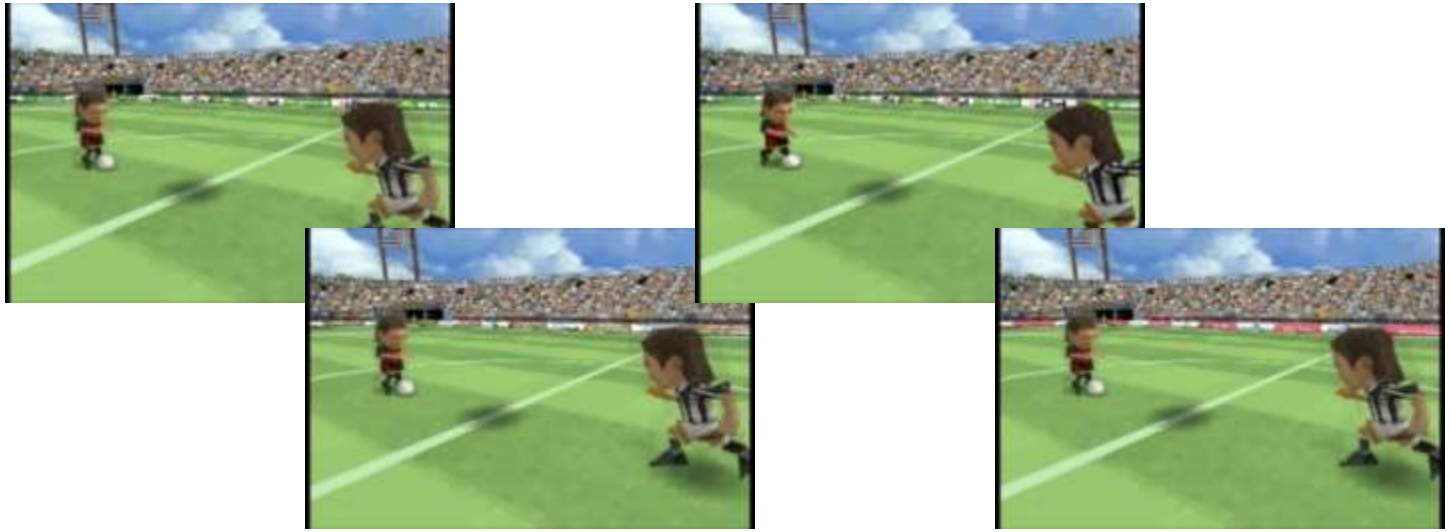
- グラフィック設計からAI設計へ
  - ボールをめぐる一連のシーンが決まれば試合の結果も決まる

シーン選択処理と試合結果判定が
直接結び付くシステムを設計しよう！

プレイセットという概念の誕生

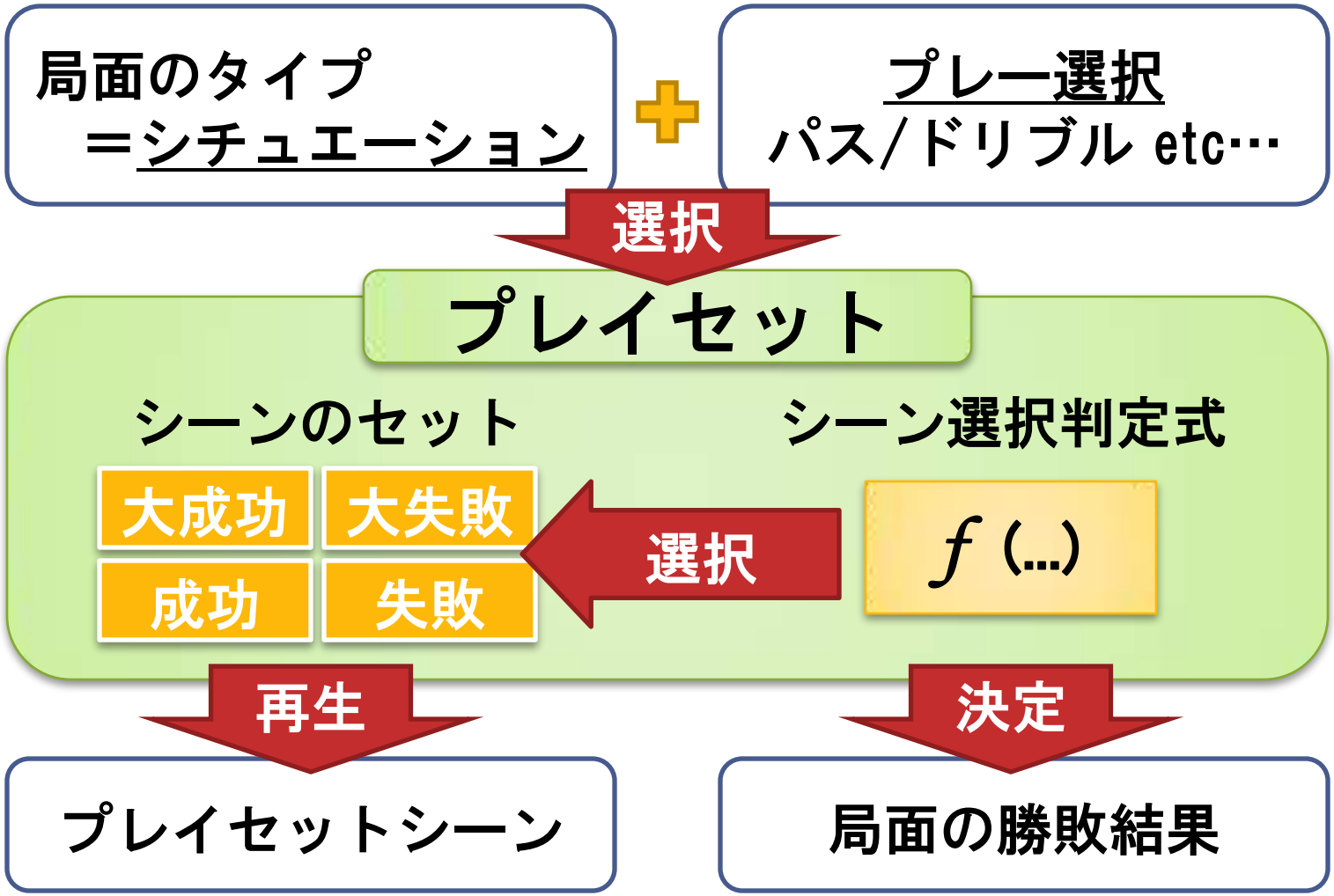
プレイセットとは？

- 同一局面、同一プレー選択から分岐し得るシーンのセット = プレイセットシーン



- 各プレイセットは、大成功～大失敗まで４パターンのプレイセットシーンで構成

プレイセットとは？


局面のタイプ
＝シチュエーション
プレー選択
パス/ドリブル etc…
選択
プレイセット
シーンのセット
シーン選択判定式
大成功
大失敗
成功
失敗
選択
f (…)
再生
決定
プレイセットシーン
局面の勝敗結果

局面計算の流れ

シチュエーション設定

↓

プレイセット選択AI

↓

プレイセット結果判定

↓

キャラクター挙動決定

局面計算の流れ【１】
～シチュエーション設定～

- シチュエーションとは

  ＝プレイセットシーンの初期状態の分類

  - オフェンス選手とボールの関係

ドリブル中？

パス受け中？

  - オフェンスとディフェンス選手の位置関係

横からDF？

正面からDF？

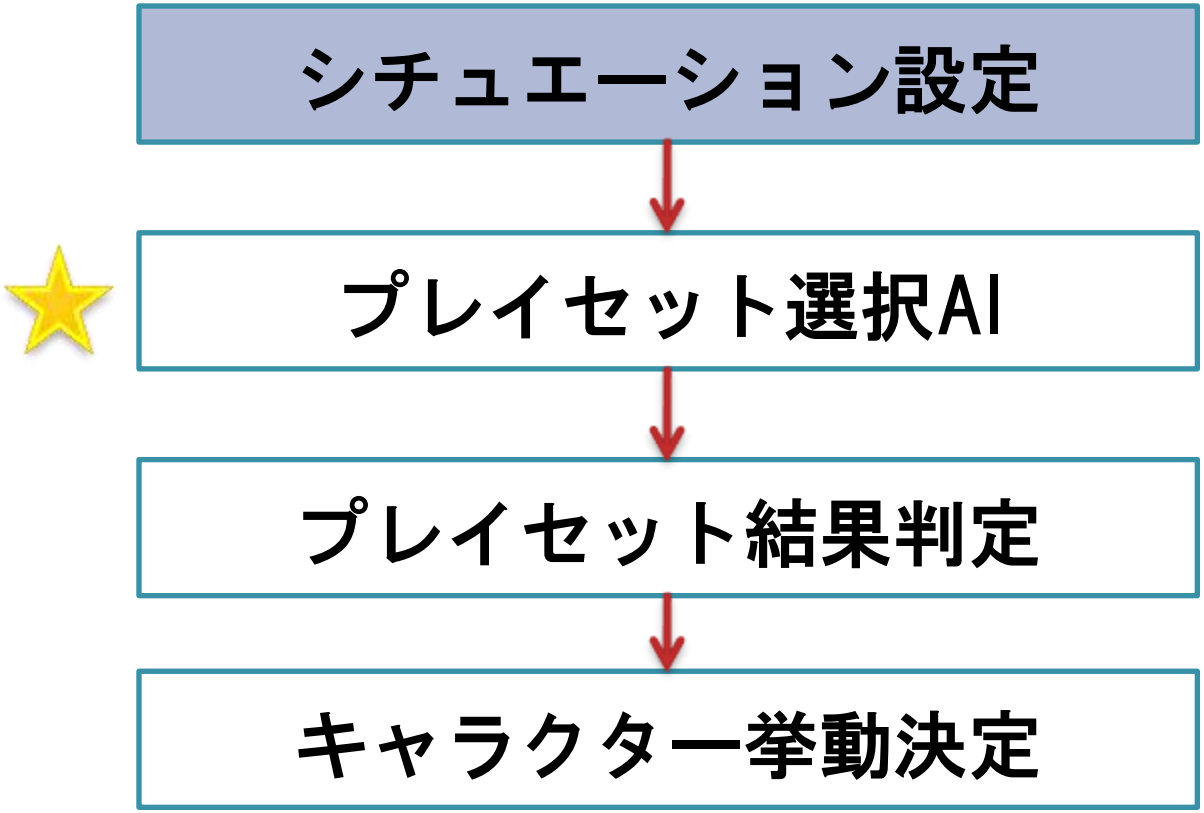
フリー？

局面計算の流れ【１】
～シチュエーション設定～

- シチュエーションを決める材料
  1. 試合コンテキスト情報
     ＝
     前回の局面の終了状態から自動的に決まる
     今回の局面の開始状態
     - 試合時刻
     - 今回のプレーが行われる地点
     - オフェンス選手とボールの位置関係　etc…
  2. チーム戦術思考
     - チームのフォーメーション
     - 守備フェーズ（自陣/中盤/ゴール前/etc…）

       ⇒ ディフェンス選手の位置取りが決まる

局面計算の流れ



局面計算の流れ【2】
～ プレイセット選択AI ～

- プレイセット選択肢リスト生成
  - オフェンス選手のプレー選択肢だけを考慮
  - 各選択肢をプレー種別＋パラメータとして記述

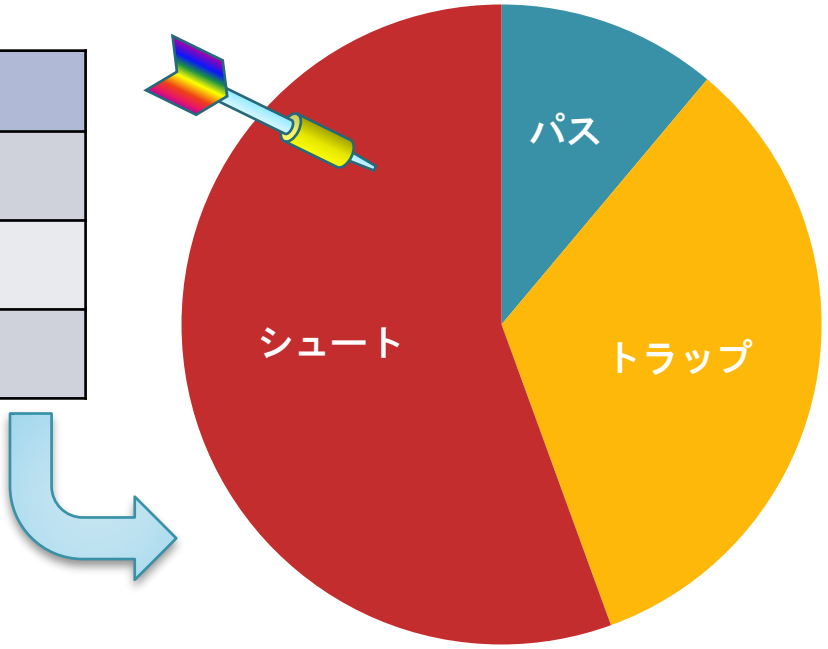
パス/ドリブル/シュート/etc…　＝　プレー種別

•パス→相手選手
•ドリブル/トラップ→方向　＝　パラメータ

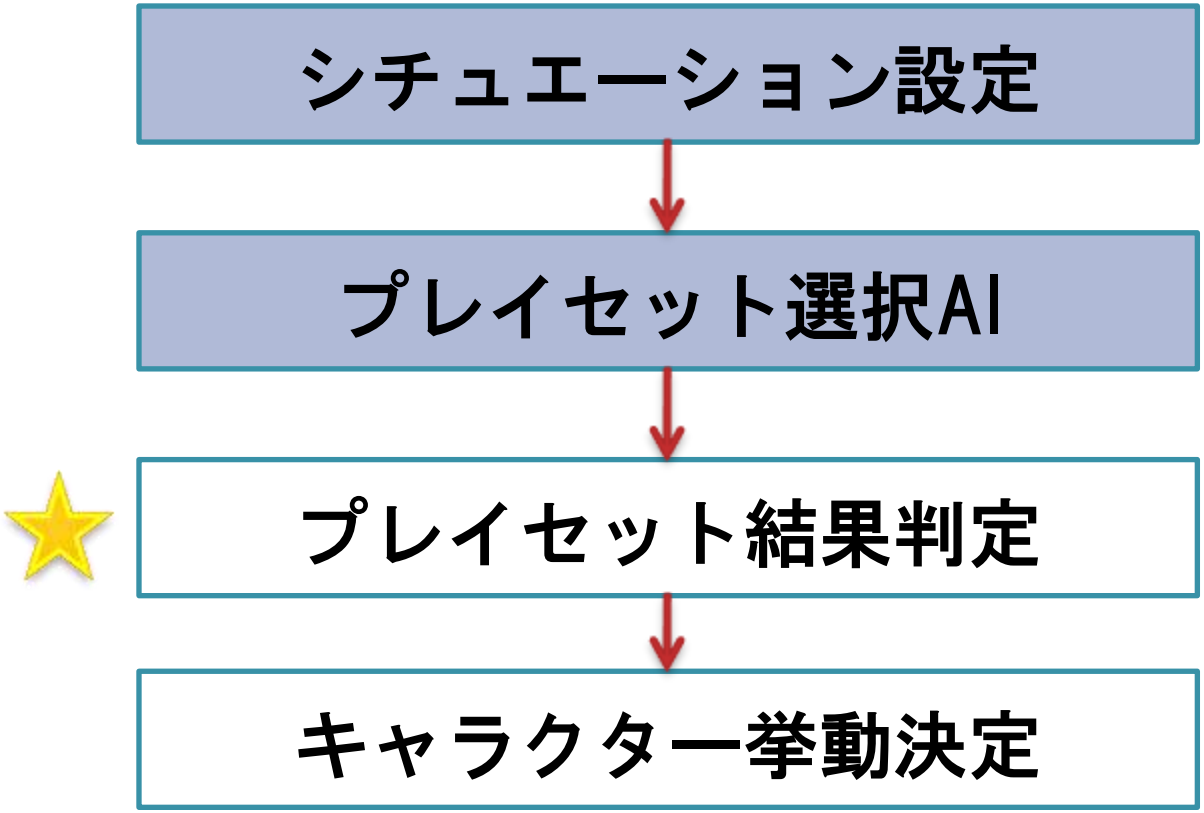
局面計算の流れ【２】
～ プレイセット選択AI ～

- 評価点方式によるプレイセット選択思考
  - プレー候補ごとの評価点を計算
  - 評価点を重みとしたルーレット方式

プレー	
選手Aへショートパス	
前方向にトラップ	
シュート	



局面計算の流れ



局面計算の流れ【３】
～ プレイセット結果判定 ～

- プレイセット固有の選択判定式から<u>確率パラメータ</u>を計算

ディフェンス

選手B
タックル=50
パスカット=40
：

オフェンス

選手A
ドリブル=40
パス=60
：

確率パラメータ $\mathcal{P}$

$= f$ (OF能力値A, OF能力値B, … ,
DF能力値A, DF能力値B, …)

※$\mathcal{P}$ の値が大きいほどオフェンス優位

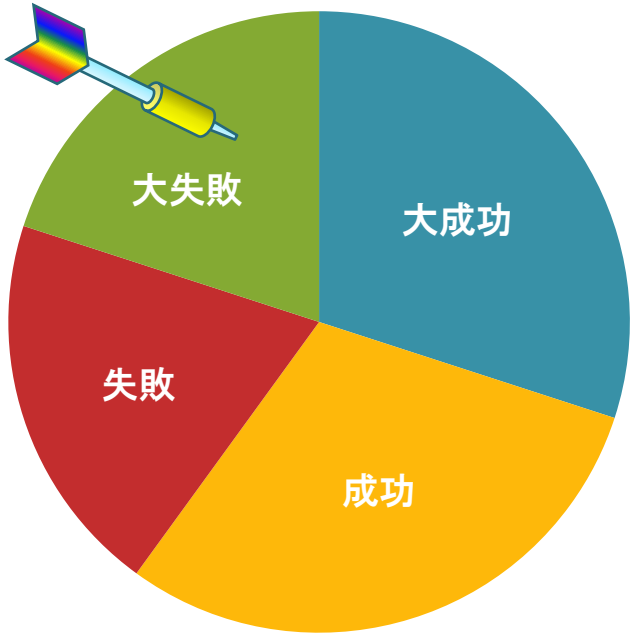
局面計算の流れ【３】
～ プレイセット結果判定 ～

- 確率パラメータを使って
  成功・失敗の確率テーブル決定
- ルーレット方式で結果判定

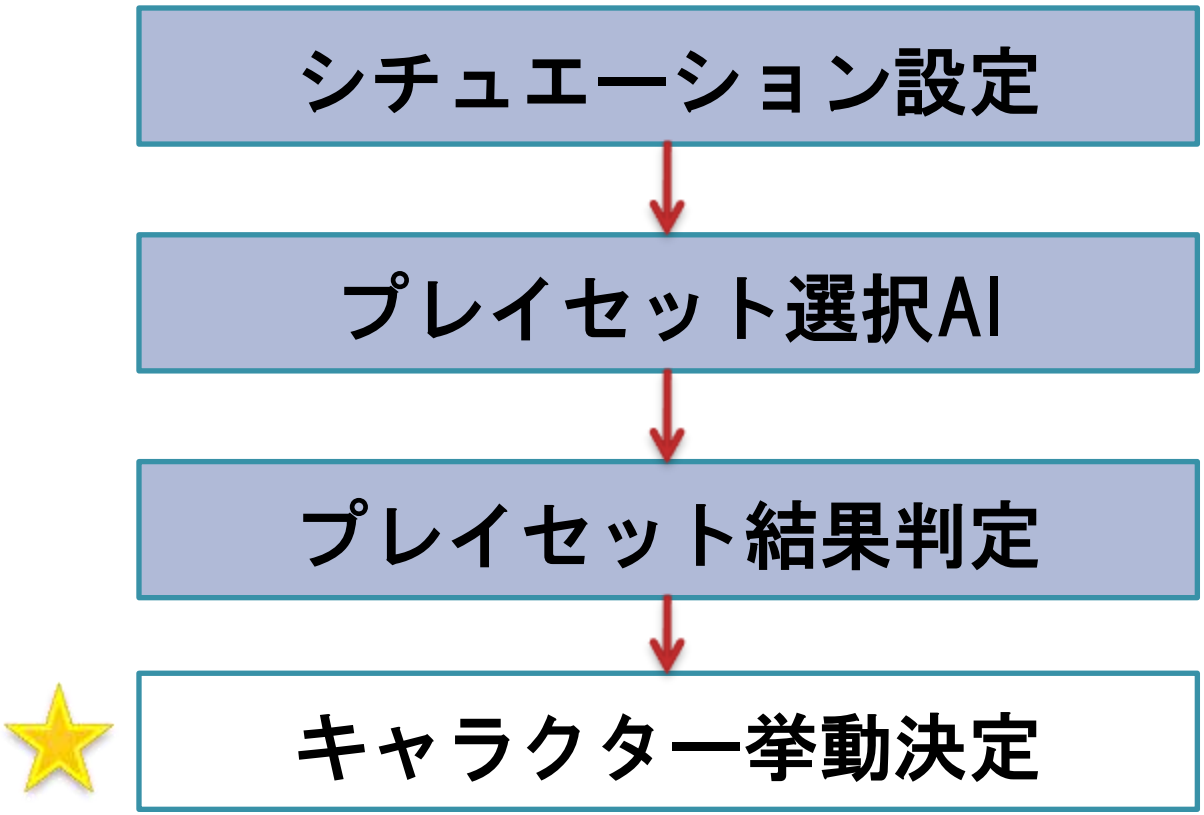
確率パラメータ $\mathcal{P}$= 55

確率テーブル

$\mathcal{P}$	30～50	50～70	70～90
大成功	5%	30%	40%
成功	10%	30%	30%
失敗	30%	20%	20%
大失敗	55%	20%	10%



局面計算の流れ



局面計算の流れ【４】
～ キャラクター挙動決定 ～

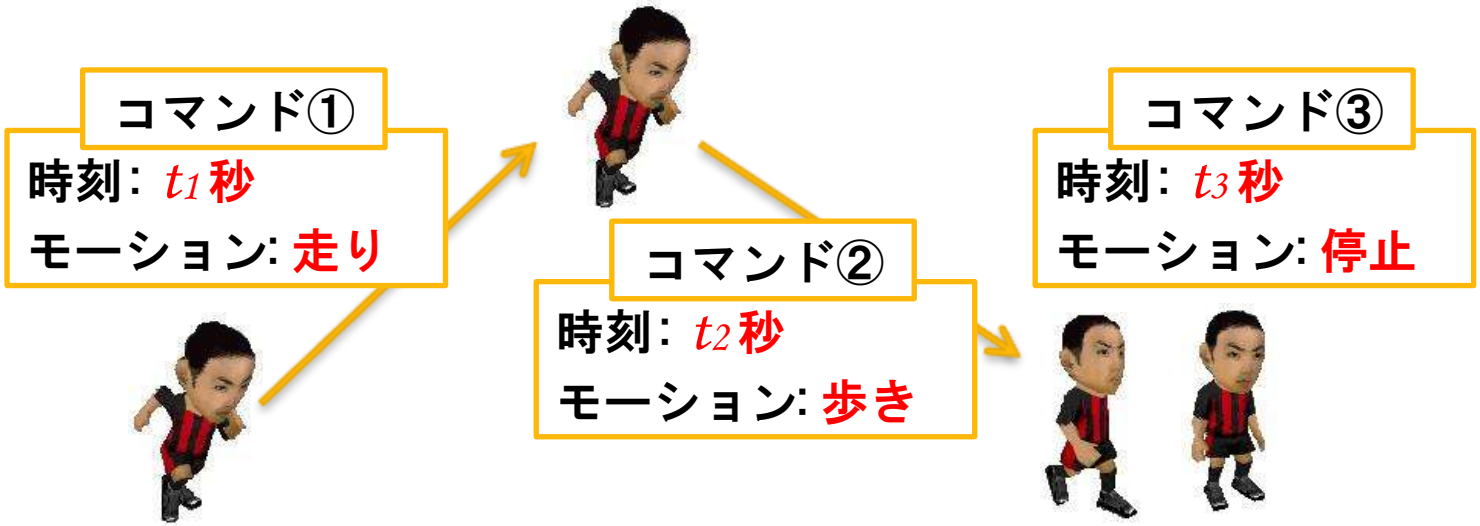
- プレイセット結果から主要部の状況が決定
  - 再生するプレイセットシーン
  - 再生位置・方向の確定
  - 再生開始タイミング調整

    ‖

    プレイセットシーン開始位置に
    選手・ボールが到達するまでの待ち時間

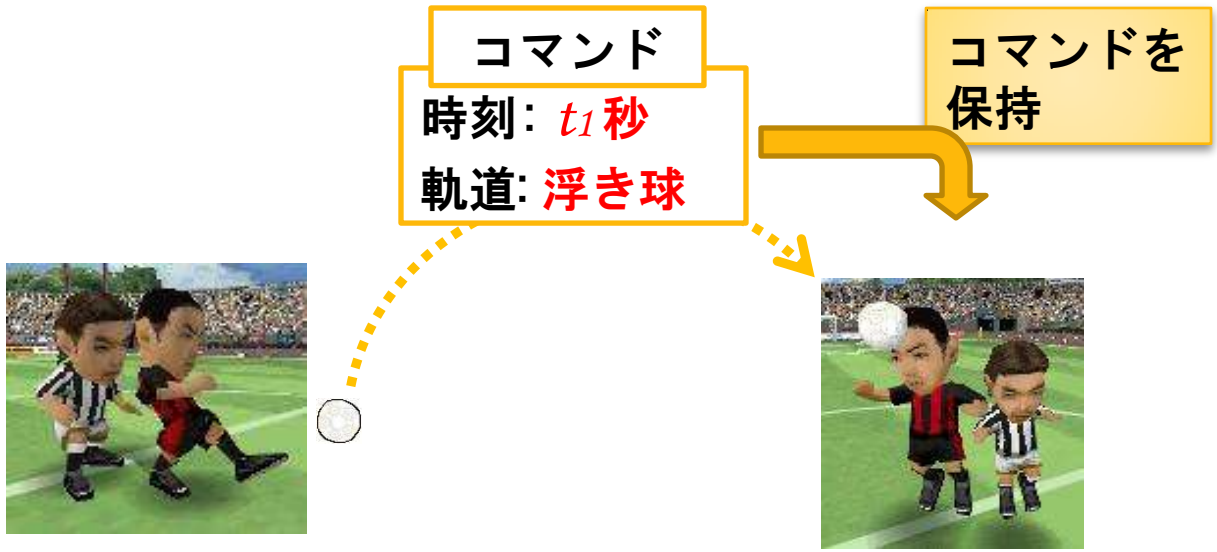
局面計算の流れ【４】
～ キャラクター挙動決定 ～

- プレイセット対象以外の選手
  - 時系列の**動作コマンド**を生成
    目標地点、使用するモーションなど

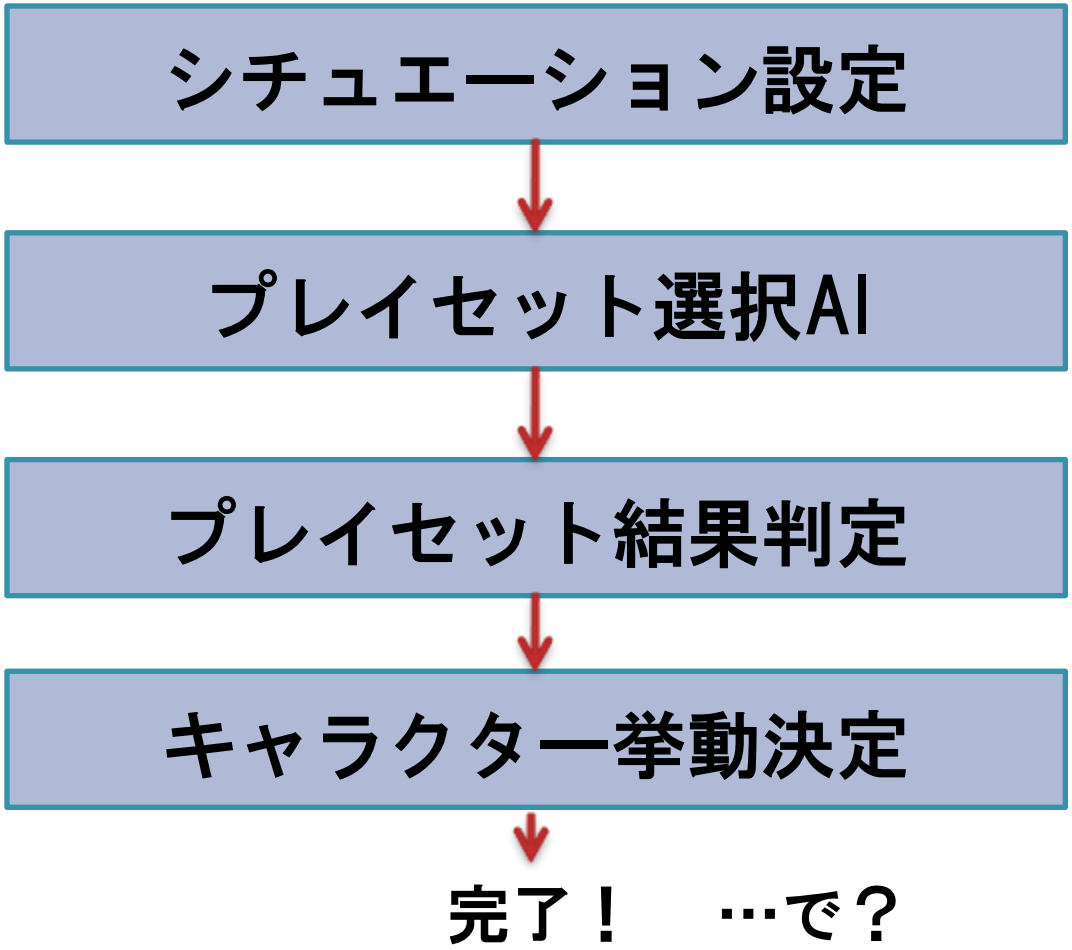


局面計算の流れ【４】
～ キャラクター挙動決定 ～

- ボール軌道
  - 前のシーンからのつなぎで
    ボールモーションがない時間帯の軌道を
    コマンド生成



局面計算の流れ



局面計算の結果管理

局面計算で計算してきたもの

プレイセット
シーン

- AI情報
- 選手ステータス
- etc…

時間

プレイセット
シーン以外の
選手たち
(動作コマンド)

- 局面計算の結果は、すべてまとめて
  １個の構造体に保存

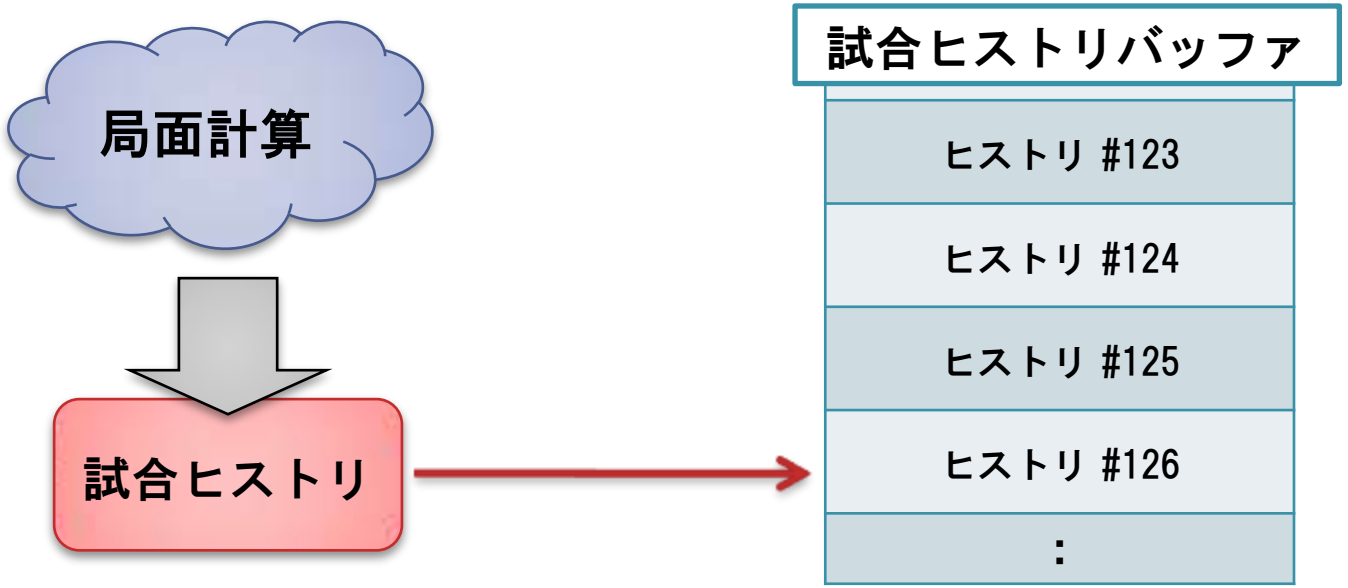
＝

試合ヒストリ構造体

局面計算の結果管理

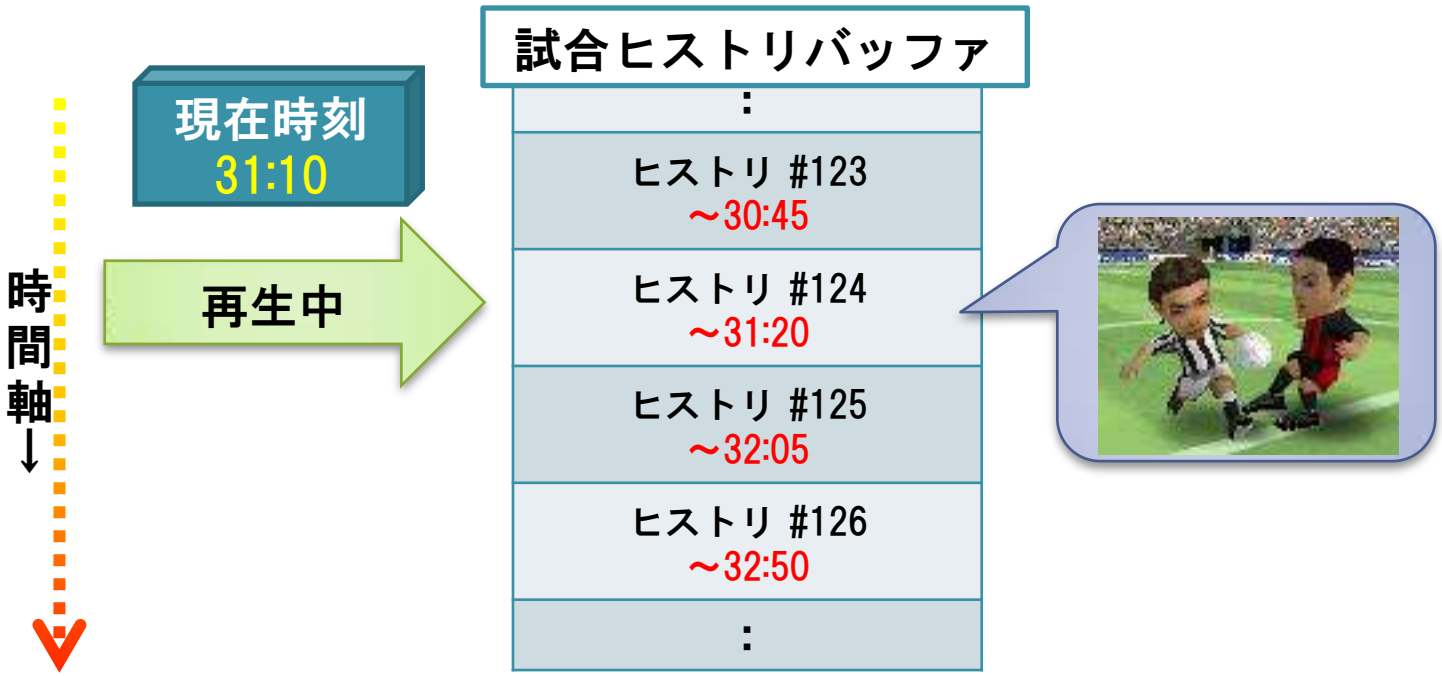
- 生成された試合ヒストリ構造体は<u>試合ヒストリバッファ</u>に保存

あとでグラフィック再生する際に参照



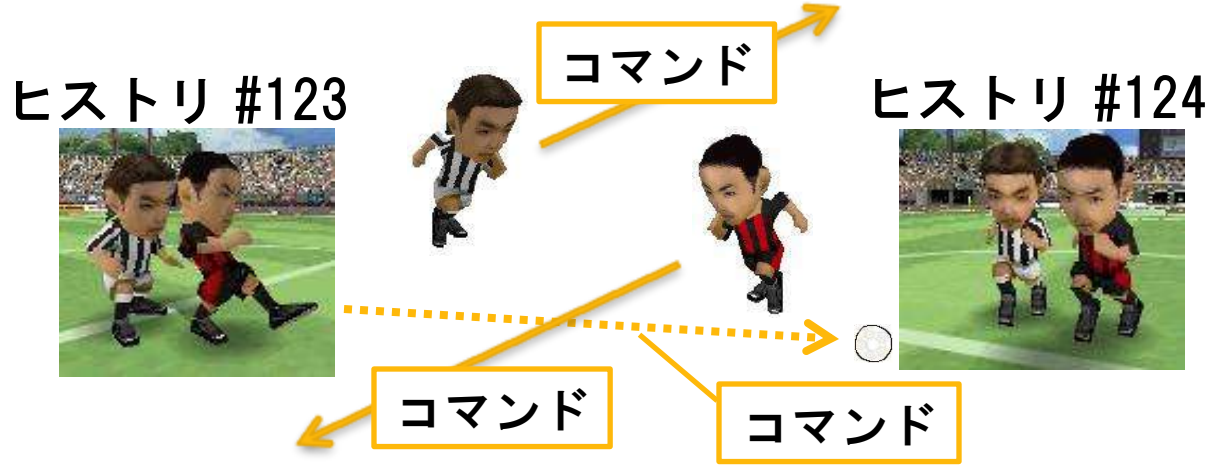
グラフィック再生

- 試合ヒストリバッファから
  現在時刻の情報を取り出して再生



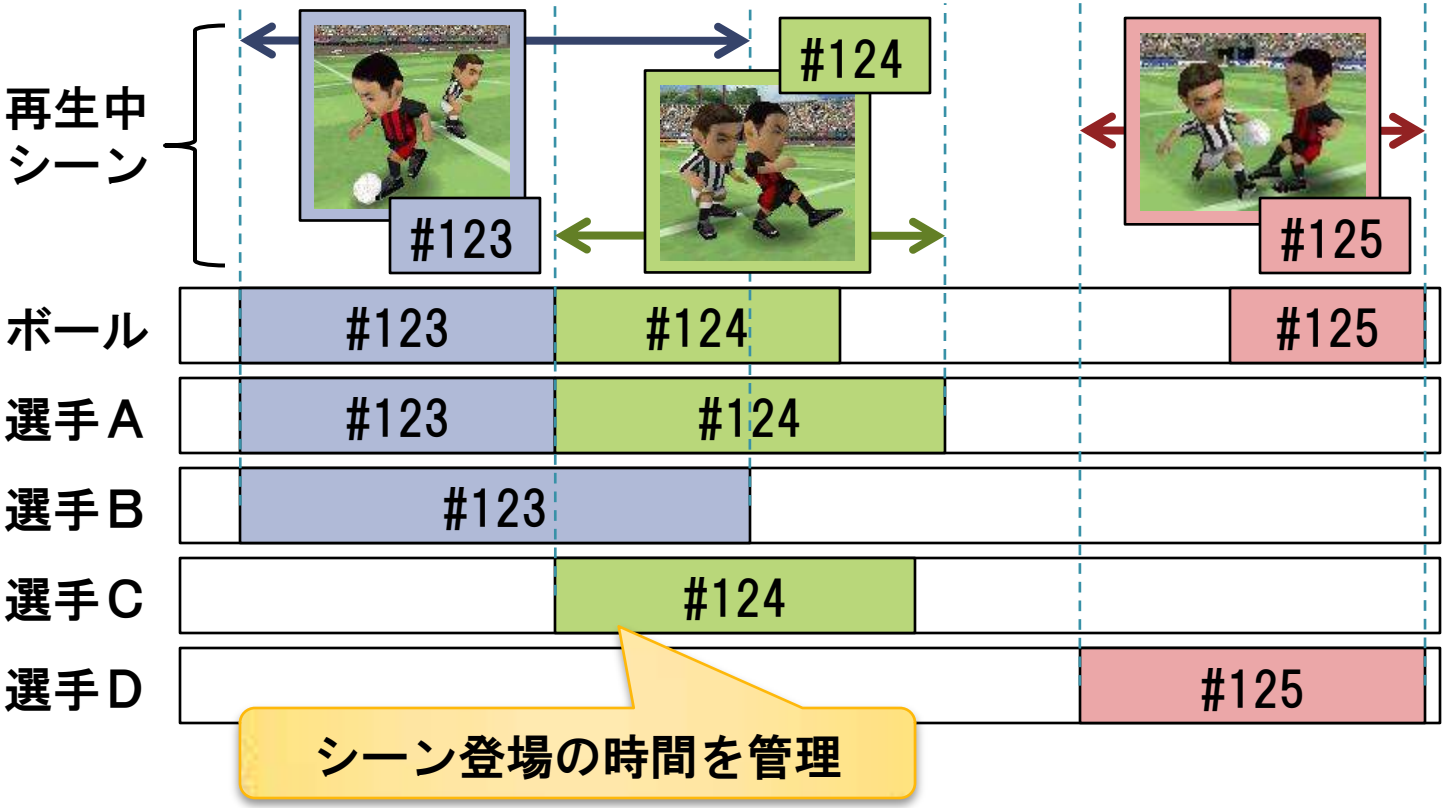
グラフィック再生

- プレイセットに登場する選手・ボール
  - プレイセットシーン再生
- その他の選手・ボール
  - 時系列の動作コマンドに従って動作



グラフィック再生

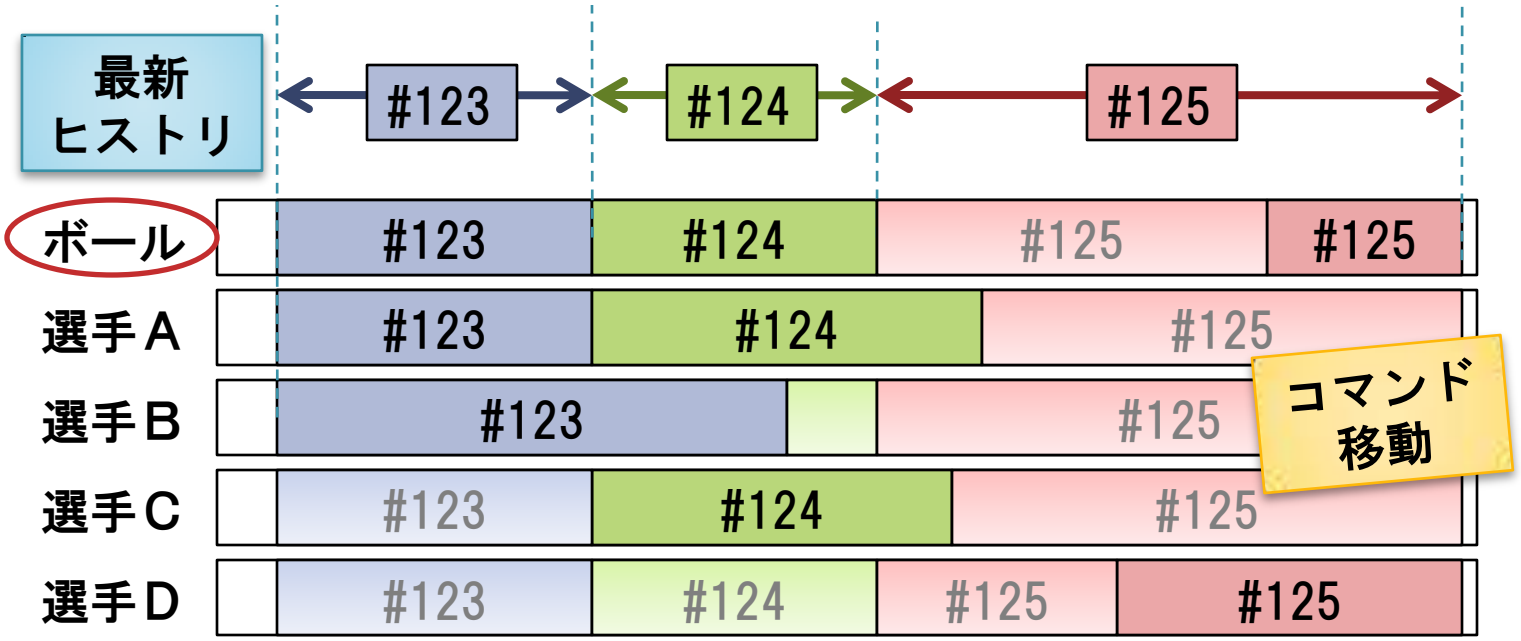
- プレイセットシーン再生は非同期
- オブジェクトごとに所属シーンを管理



グラフィック再生

- プレイセット以外の選手の移動
  - 最新試合ヒストリの動作コマンドに従って移動

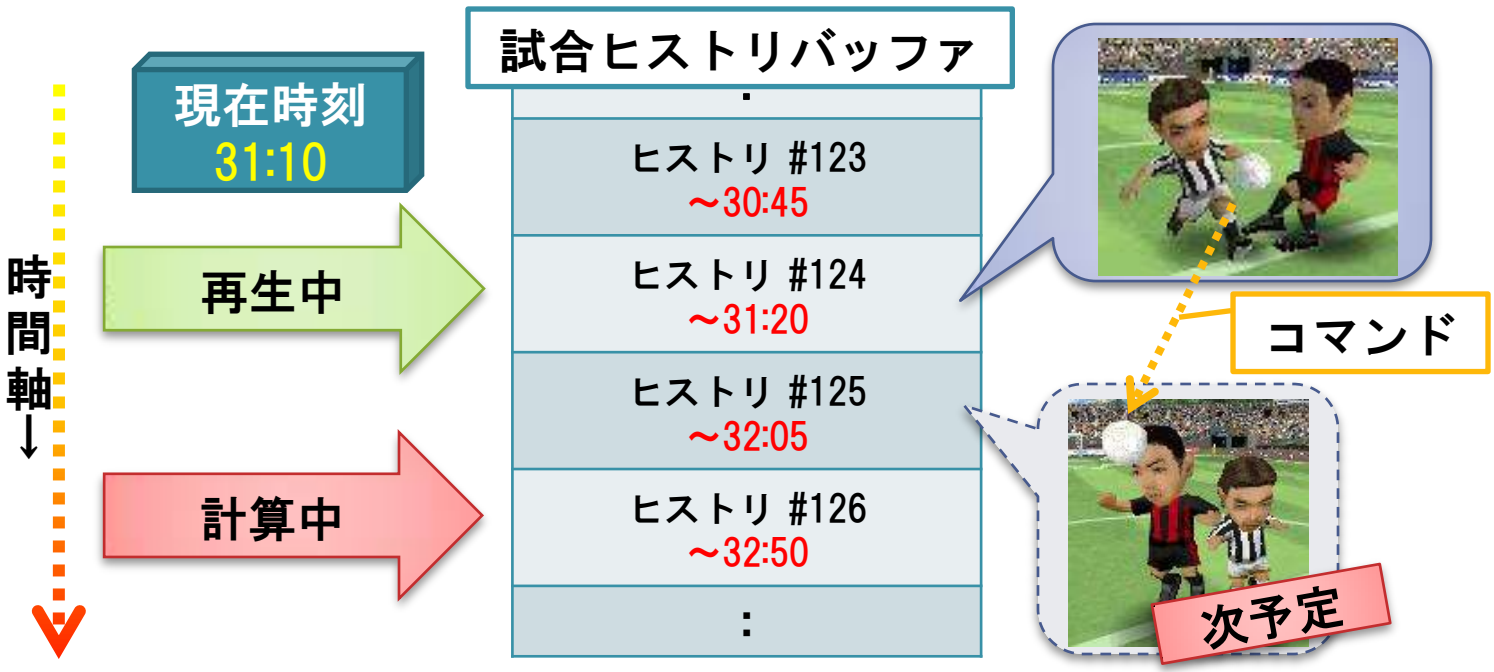
＝ ボールの軌道変化で切り替え



グラフィック再生

- ボール軌道などの動作コマンド再生は
  プレイセットシーン再生より早く始まる

常に２～３手先の試合ヒストリを計算



本章のまとめ

- コンセプト“球際だけでも美しく”
  - プレイセットの概念が誕生
  - すべての要素をプレイセットの単位で管理する試合ヒストリ構造体
- １局面の構成要素（＝試合ヒストリ）の計算処理流れ
- 試合ヒストリの配列を利用したグラフィック再生のしくみ

問題は克服されたか？

問題１：シーン制作工数
問題２：結果コントロール
問題３：実行速度
デバッグ・調整における利点

問題１：シーン制作工数

完全オーサリング方式

- 登場人物全員のオーサリングが必要
- １つの局面からの分岐数が多い

折衷方式

- シーン登場人物を限定
- １つの局面からの分岐数も限定

	完全オーサリング方式	折衷方式
登場人数	2～10数人	2人
分岐数	2～10数個	4個

シーン個数を 1/10 以下に抑制

共通化で
さらに削減

※ 完全リアルタイム方式で必要な
モーション数と同程度

問題１：シーン制作工数

- 周りの選手のモーションは？
  - ボールに絡まない動きに限定されている
  - DS版ではモーション遷移も考慮しない

モーション数は極限まで少なく

工数で言えば数人日程度

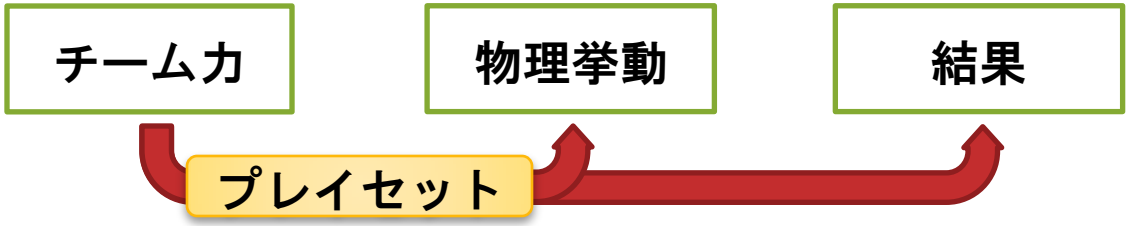
プロジェクト固有の判断

◆処理負荷の軽減優先
◆球際のグラフィックスに特化

問題２：結果コントロール

完全リアルタイム方式

折衷方式



結果のバランス調整をしやすくなった

問題３：実行速度

- 完全リアルタイム方式で、負荷の高い処理は何だったのか？
  1. モーション計算
  2. AIの状況判断
     - スペース判定
     - パスコース生成
     - ボールの軌道予測

問題３：実行速度

1. モーション計算の速度向上
   - ある意味どうしようもない
     - LOD対応など
   - AI計算をモーション計算に依存させない
     - プレイセットシーンデータからAI計算に必要な情報をあらかじめ抽出しておく
       - モーション開始・終了座標（選手・ボール）
       - モーション開始・終了フレーム（選手・ボール）

実際のモーション計算をすることなく
結果の状態を計算できる

問題３：実行速度

2. 状況判断の速度向上
   - A) 処理のチューニング
     - 確実な速度向上を見込むのは困難
     - 費用対効果が薄い
   - B) 計算頻度の削減
     - 判断の精度が下がる恐れ
     - レスポンスとのトレードオフ

試合ヒストリ単位で
問題なし！

プロジェクト固有の判断

◆“ほぼ”リアルタイムの処理分岐でOK
◆それより「高速試合」のスピードが重要

デバッグ・調整における利点

- 試合ヒストリ方式のメリット

  ピッチ上で起きることはすべて
  試合ヒストリに記述されている。

  - バグ再現性の向上
    - 構造体の退避／復帰で再現可能
  - 試合ログの減量
    - バグの発生する“一瞬”を特定しやすい
  - 超高速に試合を繰り返せる
    - バランス調整で統計に活用

デバッグ・調整における利点

- 試合ヒストリ計算実装の注意点
  - 同じ構造体の中に、計算済みメンバーと未計算のメンバーが混在

処理フェーズごとに計算済み・未計算のメンバーを把握しやすい設計が必要

- 試合ヒストリ構造体を、処理フェーズごとに分割した小構造体の集合体として定義
- 関数の引数は原則として小構造体単位
- 値参照用の小構造体は const 引数にして計算中の小構造体と明確に分ける

本章のまとめ

		完全オーサリング方式	完全リアルタイム方式	折衷方式
見た目	局所	◎	△	○
	全体	◎	○	×
展開の多様性		×	◎	○
結果コントロール		◎	×	○
制作工数		×	○	○
操作レスポンス		×	◎	△
実行速度		◎	×	○

- 見た目、結果重視なら完全オーサリング方式
- レスポンス重視なら完全リアルタイム方式
- 「サカつく」向けなら欠点が少ないのが折衷方式

総合レベルでの最適解

CEDEC 2010 CESA Developers Conference

×

サカつく.com www.sakatuku.com

おわりに

システムの特長について考察
本公演のまとめ
オマケ

システムの特長について考察

- システムの要素を再分解してみると

シーケンシャルな
AI処理 ＋ **非同期**の
シーン再生

※ 同期型のシーン再生なら、ここまで複雑なシステムにする必要はない。

- 多数のキャラが非同期にからみ合うような表現で力を発揮
- レスポンスがそれほど重要でないアクション性のあるゲーム向き

システムの特長について考察

- 他ジャンルへの応用可能性は？
  1. パーティー型のアクションRPG？
  2. バトルロイヤルのシミュレーションゲーム？
  3. 格闘シミュレーション？
     （身体の複数パーツを非同期に使う感じ）

本公演のまとめ

- 本公演で話したこと
  - 「サカつく」に求められる試合AIの要件
  - シリーズごとの試合AIシステムの実装方法と、それぞれの利点・問題点
  - 新システム設計のコンセプト
  - 試合AIシステムの処理流れ解説
  - 新システムがどのようにして問題を克服したか
  - 他ジャンルゲームへの応用の可能性

オマケ

「サカつくDS　ワールドチャレンジ2010」で
南アW杯のリベンジ！

“日本 vs パラグアイ”　50試合

日本代表　23勝27敗
返り討ちに遭いました(T-T)

ＰＫ戦
７勝３敗

「サカつくDS　ワールドチャレンジ2010」
絶賛発売中です。

ぜひ皆さんの力で
リベンジを!!!!!

CEDEC 2010 CESA Developers Conference × さかつく www.sakatuku.com

質疑応答

連絡先はこちら。

安藤 毅（あんどう・たけし）
mail: Ando Takeshi@sega.co.jp
Twitter: taco2001

本公演に関するTwitterハッシュタグ　#sakatukuAI