オブジェクト指向モデルによるサッカー競技のシミュレーション

Simulation of soccer-play with object oriented model

( ) ( )

1．はじめに

サッカーとは,「11 人を超えない競技者からなる 2 つのチームによって行われ，競技中に得点の多かったチームを勝ちとする競技」である[1]．チームが機能するためには，1 人 1 人がポジションに応じてその役割を果たす必要がある．特にスペースの少ない近代サッカーにおいては，このような戦術が勝敗を大きく左右する．しかし，戦術的な面を考慮しないならば，各選手はパスやドリブルなどの基本的な動作を行うという点で全選手が同じ行動をとるとみなすことができる．

本研究では，サッカー選手をひとつのオブジェクトとしてモデル化する．行動の点で全選手を同じとみなしうるので，選手クラスが 1 つあれば，あとは必要な人数分のインスタンスを生成すればよいことになる．したがって，サッカー選手クラス設計が本研究の大部分を占める．

ただ，競技時の人数さらにはその中での敵味方の人数によって，サッカー選手の動作や判断の数が変わってくると考えられる．そのため，11 人対 11 人での競技の前にまず 3 人対 3 人程度での競技をシミュレーションすることを目標とした．そして，選手 2 人(味方同士)から選手 6 人(3 人対 3 人)へと 1 人づつ段階的に人数を増やしていき，それに応じて動作と判断も追加していくという手順をとった．

以下では，選手 2 人(味方同士)の場合におけるサッカー選手のモデルについて述べる．さらに，Java 言語を用いて実装し，シミュレーションした結果についても述べる．

2．モデル化

2．1　サッカー選手のモデル化

選手 2 人(味方同士)の場合に考えられる各選手の行動として

・味方の位置とボールの位置の確認
・ボールのない時の動き
・二人の間でのパス交換
・ドリブル

の 4 種類が挙げられる．各選手のこれらの行動を動作とそれにいたる判断という観点から Fig. 1 にまとめた．このうちすべての基本となる行動は，味方の位置とボールの位置の確認である．選手は競技の中で，常に味方の位置とボールの位置を確認することによって各判断をくだしている．そして各選手は，このような判断に基づきパス，ドリブル，トラップ，移動の中から 1 つの動作を選択し実行する．

2．2　ボールのモデル化

本研究では，選手のほかにボールもモデル化する．ボールのモデル化に際しては，選手の人数による影響を受けることはない．

まず，ボールが動き出すという現象を考える．ボールが移動するためには，選手がボールを蹴る必要がある．選手が「ボールを蹴る」とは，蹴る目標とある程度の速さをボールに与えるということである．選手から見れば，ドリブルやパスは別々の行動であるが，ボールから見ればどちらも目標と速さを与えられるという点で同じことである．このことから，ボールの移動とは「目標」と「速さ」でモデル化される．この点についても Fig. 1 の右側にまとめた．

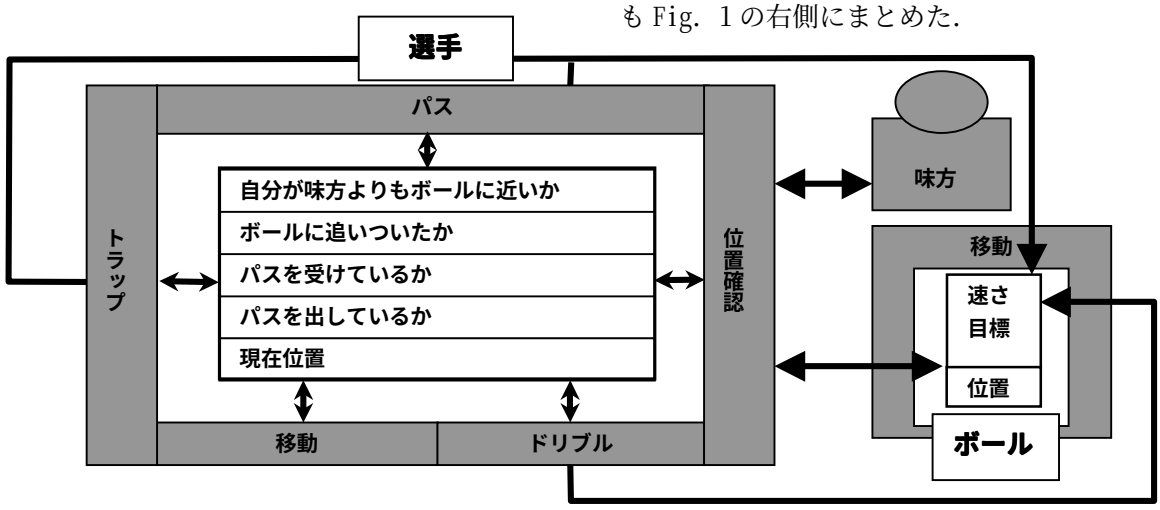

Fig. 1　選手モデルとボールモデル

次に，移動状態にあるボールは必ず静止するという性質を考える．ボールが静止する原因として，
・ボールが自然に止まる
・選手がボールを止める
の 2 種類を考えた．

まず，「自然に止まる」は，ボールを蹴る際に選手がメッセージとして味方の現在地(目標座標)，自分と味方との距離，およびボールの速さを渡す．それらを受け取ったボール側では，距離が 0 になるまで移動しつづける．こうして，ボールは目標座標付近で自然に止まることになる．この過程を Fig. 2 に模式的に示す．

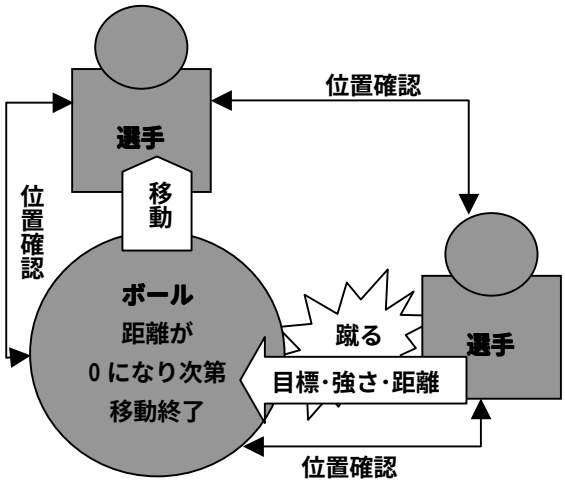

Fig. 2　ボールの移動

一方「選手がボールを止める」は「トラップ」と呼ばれる．選手は常にボールの位置を把握しており，自分の付近を通過するボールをトラップする．今回のモデルでは選手が動いているボールの目標を変えて強制的に静止状態にすることでトラップを実現している．

3．Java による実装

前節でモデル化した選手とボールに対して，オブジェクト指向言語の 1 つである Java を用いて実装した．定義したクラスは以下の 3 種類である．

・グラウンドクラス(Ground)
・選手クラス(Soccerman)
・ボールクラス(Ball5)

また，各選手とボールに対して並列的に処理可能なスレッドを適用した．このことにより，各選手が独立に，さらには非同期的に行動することを忠実に表現できる(Fig. 3 参照)．

グラウンドクラスはスレッドの生成と表示を行う．現段階では，選手クラスのインスタンス 2 つとボールクラスのインスタンスが 1 つ生成される．選手クラスとボールクラスは Runnable インタフェースを実装しており，生成された各インスタンスをもとに，Thread クラスのインスタンスを生成する．そして，Thread クラスの start を呼び出し，選手とボールが動き出すこととなる[3]．

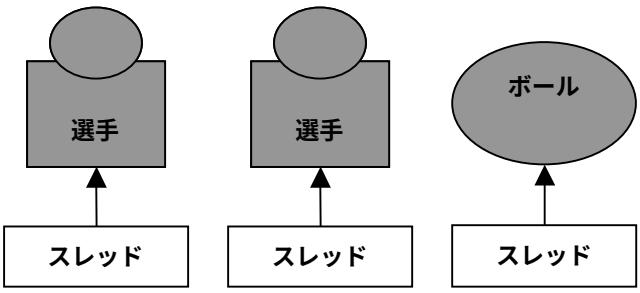

Fig. 3　インスタンスとスレッドの対応

4．まとめと今後の課題

今回のモデル化により選手 2 人(味方同士)のサッカー競技シミュレーションを行うことが可能になった．現段階では 2 人の選手がパスを繰り返しつつドリブルまたは走る様子を再現できる(Fig. 4 参照)．

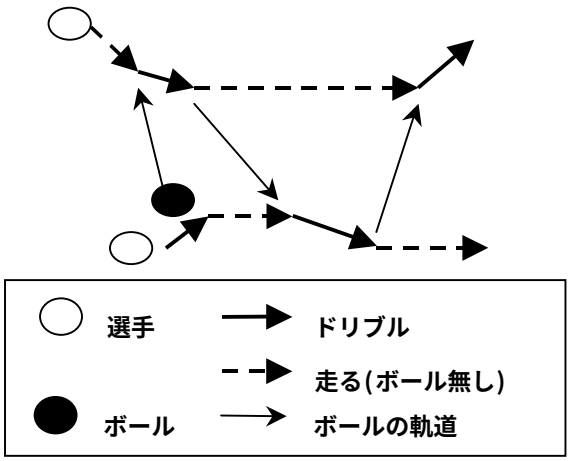

Fig. 4　競技の様子

今後，選手を 1 人づつ段階的に追加していくことになるが，それに伴い，判断･動作が増えるため，選手のモデル化がよりいっそう重要になっていくと予想される．また，オブジェクト指向の考え方や Java 言語への理解も深め，それらを応用してより忠実なモデル化を考えたい．

参考文献

[1]サッカー競技規則
http://www2.biglobe.ne.jp/~yanap/soccer/kisoku/kisoku.html(1997)
[2] 岡田謙一･重野寛･古賀祐匠：Java 教科書，ソフト･リサーチ･センター，第６章，(1999)
[3]結城　浩：Java 言語プログラミングレッスン下，ソフトバンクパブリッシング，第１６章，(1999)