AN ABSTRACT OF THE PROJECT OF

Vikram Iyer for the degree of Master of Science in Computer Science presented on December 11, 2009

Title: Interactive Methods for Authoring Behaviors of Multiple Agents

Abstract approved:

_____

Ronald A Metoyer

Virtual environments and simulations are being used increasingly to both visualize and understand data as well as to create scenarios for training and analysis purposes. In this paper, we are interested in the use of simulation and visualization of interactive virtual agents to create realistic motions for training scenarios. We explore the creation of animated scenarios for domains with complex spatial and temporal interactions to prepare or train for real possible events. Although, we are specifically studying 2D interactions exhibited in the game of American Football, the concepts and methods adopted can be extended to various fields involving spatial and temporal interactions. We present an approach for the coach to author various spatial and temporal actions by allowing him to interactively author plans, modifications, and corrections for the players' (agents') behaviors.

Interactive Methods for Authoring Behaviors of Multiple Agents

by
Vikram Iyer

A PROJECT REPORT

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented December 2009
Commencement June 2010

Master of Science project of Vikram Iyer presented on December 11, 2009.

APPROVED:

_____

Major Professor, representing Name of Major

_____

Director of the School of Electrical Engineering and Computer Science

_____

Dean of the Graduate School

I understand that my project will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my project to any reader upon request.

_____

Vikram Iyer, Author

TABLE OF CONTENTS

Topic			Page
1.	Introduction		1
	1.1	Motivation	1
	1.2	Overview	2
2.	Related Work		4
3.	Interactive Football Playbook		7
	3.1	Background	7
	3.2	New Rules	7
4.	Interface Usability Support		11
	4.1	Filters	11
	4.2	Parameterization	12
	4.3	Customization	13
	4.4	Steering Behavior	16
	4.5	Grammar and Type Checking of Rules	18
5.	Visual Coaching		22
	5.1	Sketch To Plan	22
		5.1.1 Route Specification and Modification	22
		5.1.2 Track Rule	25
		5.1.3 Parameters Specification and Modification	26
	5.2	Sketch To Modify	26
		5.2.1 Mid Play Rules	26
		5.2.2 Implementation	27

TABLE OF CONTENTS (Continued)

Topic		Page
	5.3 Sketch To Correct	29
	5.3.1 Outcome Authoring	29
	5.3.2 Implementation	30
6.	Discussion	33
7.	Future Work	35
8.	Bibliography	42

LIST OF FIGURES

Figure		Page
1.	Inside Leverage	8
2.	Route Editing	24
3.	Mid Play Rules	28
4.	Ease-in-ease-out Curve	31
5.	Speed Modification Original Position	39
6.	Speed Modification Repositioned Position	39

LIST OF TABLES

Table		Page
1.	Valid Rules in Parallel	21
2.	Valid Rules in Sequence	21
3.	Parameter Modification Table	37

1. Introduction

1.1. Motivation

There are several application domains that require interaction and training of virtual dynamic agents. These include military training, emergency response simulation and training, evacuation, sports training etc. The domain experts in these domains typically do not possess any skills in animation, simulation or programming. They are experts in their domains like military, sports etc and have knowledge of the interactions between agents in their domains. They need a simulator where they can characterize each agent in these domains by their motion or behavior easily. They must have an organized approach to communicate the strategy for each agent and visualize the future outcome.

Along with an ability to specify the behavior and characteristics of agents, they also require an ability to edit or modify it. Goal based editing is another important feature they want. This is particularly useful when the experts may know what they want but not how to specify it. The simulator must satisfy all these needs and hide the tedious details of programming the agent behavior from the expert. The interface for the domain expert to work with has to be simple without any need to possess programming or animation knowledge. The simulator must implement the strategy for the agent behavior and provide the domain expert with a visualization of this behavior and its possible outcomes.

In this paper, we focus on the application domain of training in the game of American Football. We have chosen American Football because it contains both complex spatial and temporal interactions between agents [1], in this case, the players; and because we have convenient access to domain experts at our university. As in military training and emergency response preparation, much time and effort go into preparing for a football game. This preparation typically includes the collection and segmentation of real data of the opponent, understanding how a play evolved and the attempted strategy, the re-creation of scenarios for communicating to the players what went wrong or what was a

good play and manipulation of scenarios to explore the alternatives or the desired outcomes.

1.2. Overview

Providing tools for domain experts such as military commanders, first responders, or in our case football coaches, to create and edit content in order to quickly generate effective training scenarios is a complex task. The task is complex because these individuals, while experts in their domains, are not experts in computer simulation, 3D graphics, or animation. We consequently provide tools to make the domain experts the content creators. We try to accomplish this task by taking the example of American Football and improving on the Interactive Football Playbook developed earlier [1]. The original Interactive Football Playbook was designed specifically for coaches to author various hypothetical plays and help their players visualize how these plays evolve [1].

The users of Interactive Football Playbook had the ability to create and edit football play contents. We realized the user needs to have complete command of an agent's behavior and motion in the simulation scenario he creates. In our enhanced simulator, we allow the coach to edit the content created at any time during the animation and author spatial and temporal constraints on the players. We have added features including route editing, integration of real sensor data in a scenario and modifying the same, authoring mid play rules, editing and augmenting the past and future simulation etc.

Our basic design approach to the problem was inspired from Deep Green project and consists of three phases: Sketch to Plan, Sketch to Modify and Sketch to Correct [5]. These three phases need not necessarily follow one another in order. The Sketch to Plan allows the coach to specify rules and is basically derived from the original Interactive Football Playbook. The Sketch to Modify allows the coach to edit and manipulate the parameters and the rules when the play is in motion. The Sketch to Correct is used to

correct the behavior of any agent and is specified by constraints on an agent in time and space. The details for each are described in Section 5; Visual Coaching.

The next few sections are organized as follows. Section 2 takes a look at the related work. Section 3 gives a background of the original Interactive Football Playbook and the new rules added to the current playbook. In Section 4 we provide a detailed discussion of the usability support for the simulator interface. The topics covered include filtering rules by types and by players, customization of rules and player parameters, steering behavior of players, and the grammar of rules describing the permissible combination and sequence of rules. Section 5 is devoted to Visual Coaching. We summarize our work and generalize our approach in Section 6. We end with a discussion on the future work in Section 7.

2. Related Work

The previous work on the Interactive Football Playbook provided us with a solid platform to implement various enhancements to model authoring, parameterization, planning, decision support and optimization. The Future Work section of the original Interactive Football Playbook suggested improvements for parameterization, type checking, customization and mid play rules [1]. We have enhanced the user interface to provide the coach with various customization, type checking and filtering controls. The animation is improved to implement steering behavior and collision detection, response and avoidance.

Craig Reynolds [2] implemented steering behavior for autonomous characters to provide agents with a realistic animation and locomotion model. We have adopted this work to implement steering motion for our players in the simulator.

We have also updated the collision model of the players to incorporate collision detection, response and avoidance. Collision detection is mainly inherited from the original Interactive Football Playbook. We have incorporated the concept explained in [3] for collision response of players. For collision avoidance, we make each player an observable agent that learns from its environment if any obstacle is within a specific distance of itself.

The original Interactive Football Playbook allowed the coach to plan and implement a strategy. We intend to upgrade it, add features for mid play rules and support outcome authoring. We refer to these three phases as the Sketch to Plan, the Sketch to Modify and the Sketch to Correct. The key idea for outcome authoring was inspired from the Deep Green [5] project of DARPA. The coach may reposition a player at any time during animation if he considers the repositioned point as a strategically better position for the player. We generate a list of options that satisfy the repositioning constraint. The coach may play each option and make a choice to select the option that best matches his

requirement. The options optimize the past and the future play parameters with respect to the repositioned instance. The Coach-Trainee [10] presents a similar system to satisfy goal based constraints of articulated motion using inverse kinematics.

The Deep Green project is intended to help the US Army bird colonels to manage their 'modular brigade' battle groups. It maintains a state space graph of future possibilities and uses the ongoing operation trajectories to assess the likelihood of future states [5]. The program description for Deep Green outlines several key aspects of the system; few of which are Sketch to Plan, Automated Option Generation and Sketch to Decide amongst others. The Sketch to Plan converts commander's intent and sketch to a set of actions to be performed. The automated option generation generates options from this set of actions. The Sketch to Decide allows the commander to view the future for a choice made by him.

Our system has various similarities and differences with respect to Deep Green. Both projects address the needs of domain experts who do not possess skilled animation knowledge. In both cases, the simulator is used to communicate the plan and fill in the details. The major difference between the two projects is that while the Deep Green project generates various options and allows the user to observe the ramifications of a choice; our simulator allows the user to generate options and make an informed choice as the simulator realizes each option for the user.

We are not interested in exploring the futures but to generate the desired future behavior. The main role for our simulator is to illuminate options for the coach to perform decision making by allowing him to quickly generate various future possibilities. Our system provides the ability to seed the simulation with real trajectory data and to reproduce a scenario as closely as possible to the original one. The coach may use this as a teaching model and hence we must match his mental model or give him the ability to edit to create it.

End-User Programming of 3-D Virtual Agents [4] and End-User Strategy Programming [6] also mentioned about the need for strategy oriented programming, where virtual agents may be programmed for training purposes by domain experts. These experts are content creators and do not have prior knowledge of animation. Whereas End-User Programming of 3-D Virtual Agents only highlighted the need for strategy oriented programming, we devise a functional system for the experts to author content and strategy for the virtual interactive agents.

Many other researchers have explored various techniques to provide realistic visualizations and demonstrate compelling animated characters. In Animating Athletic Motion Planning by Example [8], a data driven memory based technique is explored to build motion sequences of characters for animation. Cohen et al [9] generate visualization of abstract agents and their locations by preprocessing with Bayesian Clustering algorithm.

3. Interactive Football Playbook

The original Interactive Football Playbook presented the coach with a visual language to conceptualize his intent of creating animated football plays using rules specified on players. The coach would create the entire animation content by first creating the formations for offense and defense, and then selecting the appropriate set of rules to be followed by each player. This section explains in detail the background of the original Interactive Football Playbook and also the new set of rules added in our project.

3.1. Background

Interactive Football Playbook provided the coaches with the ability to author football play content using notations similar to what they already use in static playbooks [1]. The user interface was designed particularly for the specific user, the coach. The interface was kept fairly simple and required mainly football domain knowledge to create formations and specify rules. The interface had support for various rules including block, avoid, pursue, cover and route.

The original project had various scopes for improvements and advancements. We discuss new rules support, user usability support, mid play rules specification and modification, and outcome authoring amongst others.

3.2. New Rules

We made three additions to the set of rules in this project; these include Leverage, Shadow, Track and Wait. These are further described in detail below:

3.2.1. Leverage

A leverage rule instructs a player to maintain an appropriate leverage with respect to another player. The valid value for a leverage is an inside leverage or an outside leverage. Also a player can maintain a leverage rule with a player from an opponent team only.

3.2.1.1. Inside Leverage

In inside leverage, player A maintaining an inside leverage with respect to an opponent player B always tries to maintain a position between player B and the center. In the following figure, player A is maintaining an inside leverage with player B. Player B is running a route from its current position to slight left and then to the right. The box in the middle is the centre.

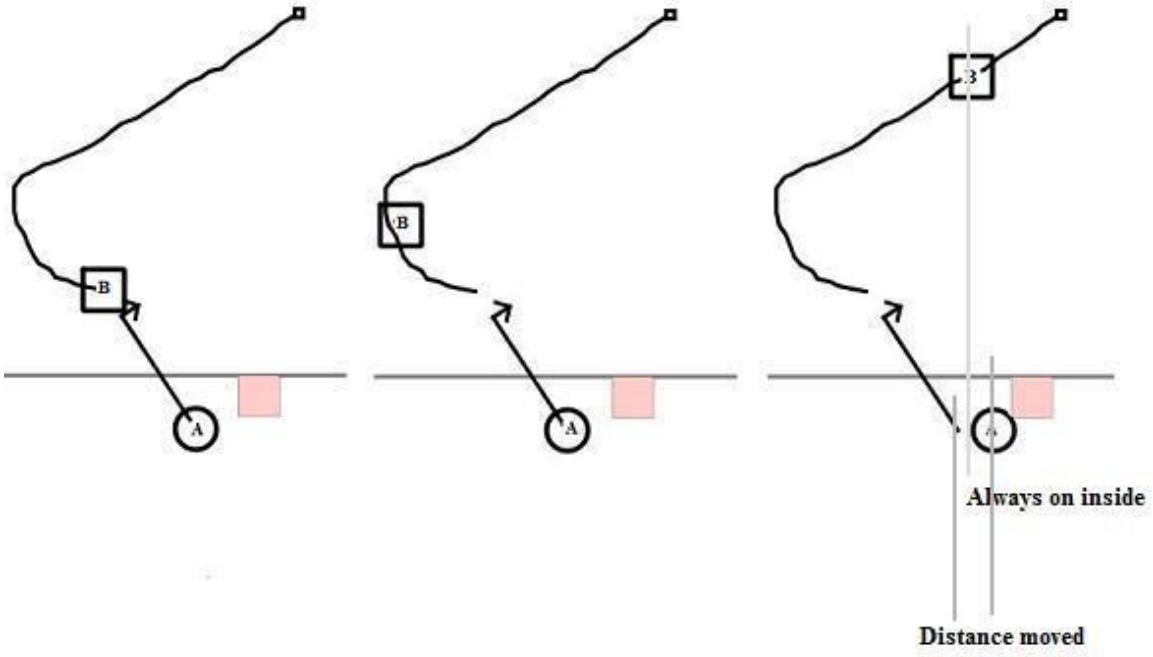


Figure 1. Inside Leverage

As player B moves to the left (outside), player A does not move since it is currently between centre line and player B (inside of B). When player B runs to the right and is about to cross player A, player A also runs to the right to continue to maintain the leverage.

3.2.1.2. Outside Leverage

In outside leverage, player A tries to position itself on the outside of player B. That is, if player B is on the left side of the centre, player A will try to position itself to the left of player B. Similarly, if player B is to the right side of the centre, player A will try to position itself to the right of player B. We say try to position because there are various factors that may not allow player A to perform its desired behavior. For example an opponent player is blocking it. For similar reasons we always say expected or desired behavior and not accurate behavior.

We allow the coach to customize the leverage parameter and its distance. This is the desired horizontal distance between the two players. Also, a leverage rule has no significance on its own. The coach has to always provide a leverage rule in parallel with a block, cover or pursue rule. The type checking in the simulator ensures that invalid combinations or sequence of rules cannot be added. This topic is covered in the Grammar and Type Checking section.

3.2.2. Shadow

The coach uses a shadow rule to specify a player A has to maintain the same vertical location on the field as another player B from the opponent team. A shadow rule is similar to a leverage rule. Whereas a leverage rule has a horizontal offset to determine inside or outside parameter, the shadow rule does not have any such parameter.

3.2.3 Wait

The wait rule specifies a player must wait and do nothing for a specified period of time. The coach may customize the time. The player is waiting for an event or for a time period. This is especially useful at the start of the play until snap. We have not explored the applications of the shadow and the wait rules in this project. The brief description here allows users to understand how to specify these rules and enhance it for their specific needs.

3.2.4 Track

Track is another important rule added to our simulator. The track rule is used mainly to simulate real football plays. Real sensor data are fed into the system and their simulations are performed via track rules. The coach cannot create or customize a track rule since it is created only from real sensor data fed into the simulator. Track rule instructs a player to position itself at specified locations at specific time instants. The track rule and its integration are explained in detail in the Visual Coaching; Sketch to Plan section.

4. Interface Usability Support

The various domain experts do not have much programming or animation experience. There is a constant need to keep the user interface simple, easy to understand and use, familiar and tailor-made for the domain experts. Our simulator is developed keeping in mind football coaches and playbooks. The interface notation, customization and parameterization options are supported to make the task easy for the coaches. The interface is kept simple without any tedious requirements to specify the entire play sequence or interactions for each player. We discuss the various interface customization and parameterization support in this section.

4.1. Filters

The filters provide the coach with an ability to diagnose a particular behavior or a particular rule usage. The set of rules may be filtered based on a particular player or a particular rule. This may be very useful in creating strategies like how many players are executing a block rule or what are the different rules that a player is performing. The coach may have certain numbers in mind for each rule, for example two players should be pursuing, four players blocking etc. Instead of counting through the various rules in the scenario or having to remember each specified rule, he may simply apply these filters and get the desired subset of rules. The various rule filters are provided as check boxes on the interface. The filters are only for viewing purposes. The animation phase will play all the rules including the filtered out rules.

There are three other check boxes for player filter, collision enable and reposition player. The player filter check box is deactivated by default and activated if a player is selected. The coach selects a player whose rules he wants to filter, checks the player filter check box and all rules except the selected player's rules are made invisible. These filters provide an easy way for the coach to analyze player behavior. The collision enable filter activates the collision detection, avoidance and response system. The reposition player

check box is used to reposition a player during animation. This check box is also deactivated by default and is activated if a player is selected during the animation phase. During animation the coach may press the pause button, select the reposition player checkbox and specify a spatial constraint on a player by dragging it to the desired location. The reposition player and collision enable check boxes are described in detail in the next few sections.

4.2. Parameterization

We identified the need to parameterize the players and the rules in the original Interactive Football Playbook. There are various parameters added for the player and for each rule. The main rules that we concentrate on are block, cover, route, wait and leverage. These parameterization options are discussed next.

4.2.1. Player

The player characteristics like player speed, weight and strength are added as parameters to the simulator. The player speed is used in determining the motion of the player. The speed and mass are used for momentum calculations and the strength is used as a parameter to determine relative strengths of two players involved in a collision.

4.2.2 Block

A block rule has two parameters; blocking time and blocking boundary angle. A player blocks another player for a specified amount of time. The blocking may be done indefinitely as well. The blocking boundary angle depends on the initial positions of the two players and is currently set as perpendicular to the initial distance between the two players. The blocking angle has been added as a parameter but is currently not customizable.

4.2.3 Cover

A cover rule instructs a player A to cover an opponent player B by maintaining a specific distance in between. The vertical distance maintained between the two players is added as a parameter to the cover rule. The horizontal distance is provided by the associated leverage rule.

4.2.4. Route

The route running rule has a parameter named route speed. This is the speed with which the player runs the route. A default value of route speed means that the route will be run with the speed of the associated player.

4.2.5. Wait

A wait rule instructs a player to wait in a specified position for a certain period of time. This period of time is a parameter specified on the wait rule.

4.2.6. Leverage

A blocking, covering or a pursuing player maintains a leverage with respect to the target player. The leverage can be inside or outside, which is specified as a parameter to the leverage rule.

4.3. Customization

We provide the coach with the ability to customize the various parameters. We use various user interface entities including sliders, radio boxes, check boxes, buttons etc. to allow the coach to customize and specify a value. Both the player and the rule parameters may be customized. The coach has to double click the player or the rule to customize the

associated parameters. A new dialog box opens where these parameters and their current values are displayed. The appropriate defaults are set for each parameter. The new values for these parameters may be customized in the dialog box and then set by pressing the 'OK' button.

4.3.1. Player

The player speed, strength and mass may be customized using sliders from the user interface. The speed is calculated from the 40 yard dash time specified by the coach. The 40 yard dash time ranges from 4.8 seconds as slowest to 4.3 seconds as fastest. The player speed is the maximum speed with which it can run a rule.

We use the equations of motion to calculate the maximum acceleration. The player accelerates from rest to top speed in 2 seconds and then maintains his speed [11]. Let's suppose the 40 yard dash time is t, the distance covered in the first 2 seconds is $s1$ and the distance covered in the remaining time $(t - 2)$ is $s2$. From equations of motion, for the first 2 seconds we have,

$$s1 = ut + \frac{1}{2}at^2$$

$$v = u + at$$

Here,

$u = 0$,

$t = 2$ seconds

Substituting these values we have

$$s1 = 2a$$

$$v = 2a \qquad (1)$$

This equation (1) gives the player's maximum speed.

For the remaining time $(t-2)$ seconds,

$$s2 = v(t-2) + \frac{1}{2}a(t-2)^2$$

Here,

$a = 0$,

Now,

$$s1 + s2 = 40$$

$$2a + 2a(t-2) = 40 \qquad (2)$$

This equation (2) gives us the value of acceleration a.

The player mass may be selected from a range of 210 to 360. The player strength may be selected from a range of 50 to 100. The strength is a relative field and is specified in percentage basis.

4.3.2. Block

After adding the block rule, the coach may customize the parameters by double clicking it. The block time may be set between 0.5 seconds and 3.0 seconds using a slider. There are markings every 0.5 seconds on the slider. The coach may select the indefinite time check box to indicate an indefinite period for the block rule. The blocking angle slider has values from 90^{O} to 270^{O} with markings every 45^{O}.

4.3.3. Cover

The cover slider has markings from 0.5 to 3.0 yards with markings after every 0.5 yards. The coach may select the distance from the slider and then press the 'OK' button to set the vertical cover distance.

4.3.4. Route

The route speed may be set between 3.0 yards/second and 8.0 yards/second. The speed slider has markings every 1.0 yard/second. The default route speed check box selection results in the player running the route with its own speed. The route speed cannot be more than the player's speed.

4.3.5. Wait

The wait time customization dialog box has a slider and a button. The slider has a range of 0.0 seconds to 5.0 seconds with markings every 1.0 seconds. The button is an 'OK' button that is pressed to set the new wait time.

4.3.6. Leverage

The leverage rule parameter may be customized by double clicking a leverage rule. A dialog box opens where the coach may select either the inside or the outside radio button and then press 'OK' button.

Although there are potentially several parameters that may be specified and customized, we have limited our analysis to only the ones specified in this paper.

4.4. Steering Behavior

We intend to provide a realistic visualization for the simulated play and the player motion. We realize that the uniform velocity model in the original Interactive Football Playbook needs to be updated to simulate a more realistic approach. We implement steering behavior for our players to fulfill these needs and adopt the work of Craig Reynolds [2].

The players always have certain characteristics associated with their motion. They start from rest with a speed of zero and may accelerate up to a maximum of their top speed, run the fastest in straight directions and slow while making sharp turns, run fast to reach a destination and slow down when they meet any obstacle etc. We implement these characteristics for our players and discuss them next.

The player starts from rest with zero velocity. Its acceleration and maximum speed are obtained from the 40 yard dash time that the coach specifies. The player accelerates from zero until it reaches its maximum speed. The player need not decelerate from its maximum speed when running in straight directions; but in curved paths, its speed is a percentage of its maximum depending on the amount of curve in its path. We calculate the angle of the curve in the player's path and associate a tolerance value with this angle. This tolerance provides the percentage of deceleration that must be applied to its maximum velocity.

When the player arrives at a point, its speed gradually decreases to zero. To implement this arrival behavior, we assume it can decelerate at twice the rate with which it can accelerate. To keep things simple, our simulator ignores friction. Suppose the current speed of the player is u. Its final speed v when it arrives at the point is 0. Its deceleration is $-2a$ where a is its acceleration. From the first equation of motion,

$$v = u + at,$$

$$t = \frac{u}{2a} \qquad (3)$$

This equation (3) provides us with the time the player will take to decelerate to zero. From the third equation of motion,

$$v^2 = u^2 + 2as,$$

$$s = \frac{u^2}{4a},$$

This is the distance from which the player needs to start decelerating to follow the arrival behavior. The new velocity *u1* is given by,

$$u1 = \frac{u * (\text{distance to arrival point})}{s}$$

The updated current velocity for the player that handles the arrival behavior is the lower of *u* and *u1*.

We have enhanced the collision system in our simulator and have added a module each to handle collision detection, collision response and collision avoidance. We assume each player occupies a small space around its current coordinate location in the simulated field. We associate a threshold between two players' coordinate locations and if the distance between any two players is less than this threshold, a collision is detected. The collision response or collision avoidance system is now activated and that determines the resultant motion of the two players. We assume if both the players have a very low velocity, lower than a threshold of 3.0, they are able to avoid a collision. But if at least one of the players has a velocity higher than the threshold, we assume the players will collide and the collision response system calculates their effective velocity. We have implemented an elastic collision in 2D and applied the technique developed in [3] to obtain the resultant velocity for our players. The resultant animation with the steering behavior and collision looks relatively natural. It required adjustment of various threshold values.

4.5. Grammar and Type Checking of Rules

The original Interactive Football Playbook assumed the coach would create only legal scenarios. There were no checks performed to verify the validity of the specified combination of rules. We have identified a grammar to denote the set of allowable rules and implemented a type checking to validate acceptable combinations of rules in our simulator. The grammar has been designed specifically for the domain of American

Football. Currently, we do not have a facility for the various domain experts to specify a grammar for their respective domains.

The grammar, G, for our simulator language to specify the set of rules is defined as follows:

$G = (V, \sum, R, S)$

Here,

G is the grammar

V is the non terminal set and consists of {S, R1, P1, P2, C1, C2, B1, B2, H1, H2}

$\sum$ is the alphabet set containing the various rules - Route(r), Block(b), Pursue(p), Avoid(a), Cover(c), Leverage(ℓ), Shadow(s), Wait(w)

R is the production rules and specified as shown below.

S is the starting symbol denoting the set of rules specified for a player

R is specified as follows:

$S \rightarrow \varepsilon$

$S \rightarrow \sum - \ell$

$S \rightarrow rS \mid bS \mid wS$

$R1 \rightarrow r \mid\mid a$

$P1 \rightarrow p \mid\mid a$

$B1 \rightarrow b \mid\mid a$

$H1 \rightarrow s \mid\mid a$

$C1 \rightarrow c \mid\mid a$

$P2 \rightarrow p \mid\mid \ell$

$B2 \rightarrow b \mid\mid \ell$

$H2 \rightarrow s \mid\mid \ell$

$C2 \rightarrow c \mid\mid \ell$

$S \rightarrow SR1 \mid SP1 \mid SP2 \mid SC1 \mid SC2 \mid SH1 \mid SH2 \mid SB1 \mid SB2$

where $\mid\mid$ denotes rules that exist in parallel

The valid set of rules can be identified from this grammar. We have implemented type checking using this grammar to ensure only valid combinations of rules are added to the simulator. These valid combinations are described in the following two tables. The rules that can execute in parallel are shown in the first table with a corresponding Y, and the rules that cannot exist in parallel as N. Similarly, for rules executing in sequence, the second table shows the valid combinations.

Table 1 describes in detail the set of rules that can exist in parallel to one another. When a player is running a route, the only thing it can do in parallel is to avoid another player. Any other rule may not only cause a motion in a totally different direction, but also impede the legality of the specified rules. Similarly, a block, pursue and cover can also have an avoid rule in parallel. Also, a leverage rule may be specified in parallel to a block, pursue or cover and must be specified subsequent to one of them. A wait rule cannot exist in parallel with any other rule. A player may shadow an opponent player and at the same time may maintain leverage with or avoid another opponent player.

The wait, block and route are the only rules that can be chained. This means that for rules executing in sequence, the former rule has to be one from this set.

Parallel	Route	Pursue	Block	Cover	Avoid	Wait	Shadow	Leverage
Route	N	N	N	N	Y	N	N	N
Pursue	N	N	N	N	Y	N	N	Y
Block	N	N	N	N	Y	N	N	Y
Cover	N	N	N	N	Y	N	N	Y
Avoid	Y	Y	Y	Y	Y	N	Y	N
Wait	N	N	N	N	N	N	N	N
Shadow	N	N	N	N	Y	N	N	Y
Leverage	N	Y	Y	Y	N	N	Y	N

Table 1. Valid Rules in Parallel

After

Before / Sequence	Route	Pursue	Block	Cover	Avoid	Wait	Shadow	Leverage
Route	Y	Y	Y	Y	Y	Y	Y	Y
Pursue	N	N	N	N	N	N	N	N
Block	Y	Y	Y	Y	Y	Y	Y	Y
Cover	N	N	N	N	N	N	N	N
Avoid	N	N	N	N	N	N	N	N
Wait	Y	Y	Y	Y	Y	Y	Y	Y
Shadow	N	N	N	N	N	N	N	N
Leverage	N	N	N	N	N	N	N	N

Table 2. Valid Rules in Sequence

5. Visual Coaching

The three phases that allow the coach to create and edit a football scenario are Sketch to Plan, Sketch to Modify and Sketch to Correct. This section describes the need and implementation details of each of these phases. The three phases do not necessarily follow one another. The Sketch to Modify and Sketch to Correct are used to modify and correct the behavior of a player respectively. While the Sketch to Plan is required to specify a scenario, the coach may or may not use the other two phases.

5.1. Sketch to Plan

The Sketch to Plan phase allows the coach to create a football scenario by specifying rules for the players. The players are in the offense and defense formations, as created by the coach. This phase is an improved version of the original Interactive Football Playbook and the following modifications have been implemented in this project.

5.1.1. Route Specification and Modification

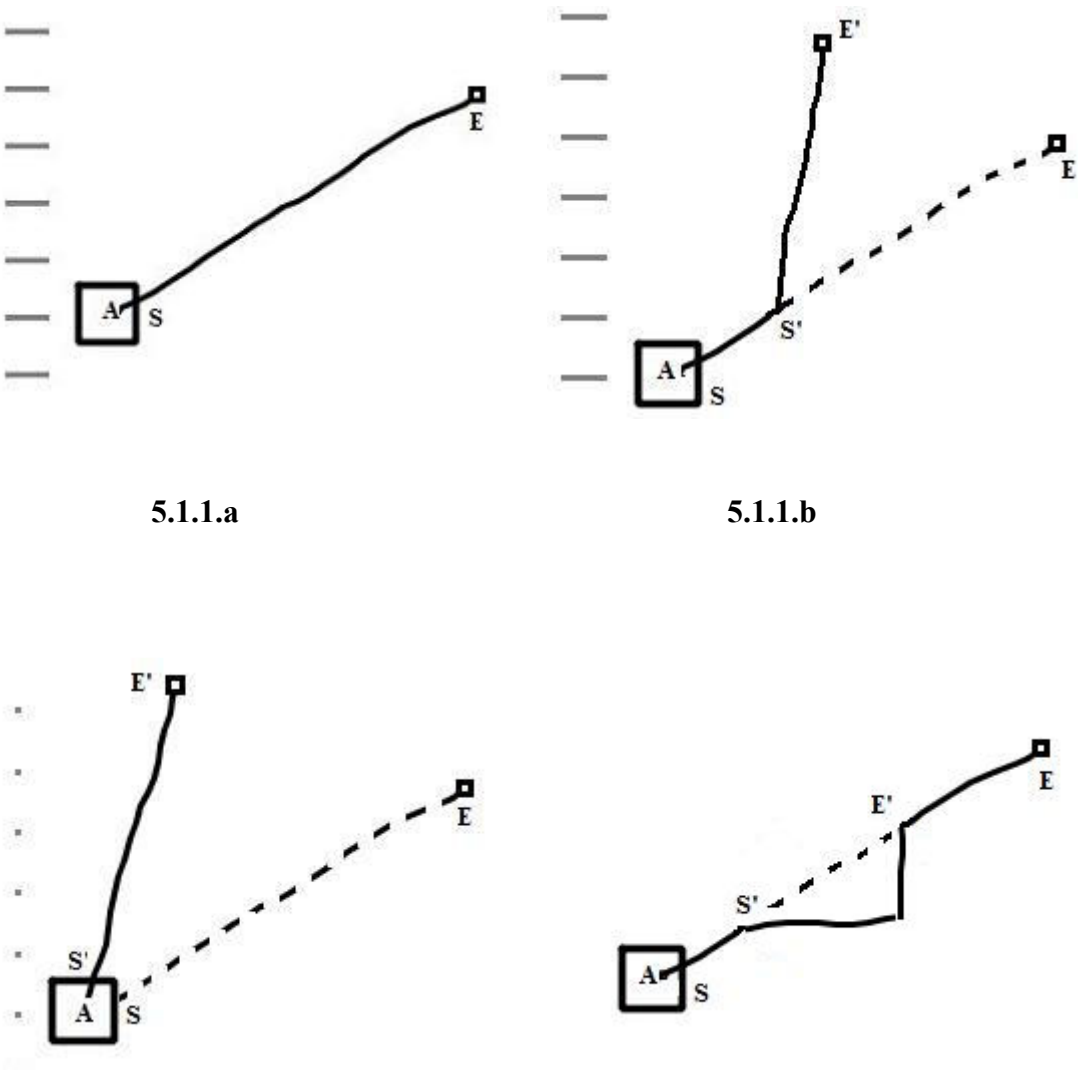
A route rule specifies a path for a player to traverse. In real football plays, players run the shortest distance from one point to another. The entire path itself may consist of many segments. In our simulator, the coach specifies the path by free hand sketches. This free hand representation may consist of various curves and it is necessary to convert it to distinct segments. We have implemented the segmented least squares algorithm [7] to satisfy this requirement.

We have also added route modification features to our simulator. Occasionally, the coach may desire to modify a specified route for a player. The coach plays the scenario with the original route and decides he wants to edit the path for the route rule. To realize this, we allow route editing to be performed in static or at run time. The static route modification

is examined in this section and the run time modifications are discussed in the Mid Play Changes section.

Route modification is performed by editing or replacing the path that constitutes the route. The coach specifies a route for a player, and then after animation desires a partially different behavior. The coach may make the necessary changes in a static manner by following a series of steps. He stops the animation and presses the 'Route' button on the simulator which is used to add a new route to a player. This button is also used to edit or replace an existing route. The coach then specifies a new path for the player to traverse. If this path is specified over the original path, the original path is modified; else the new path replaces the original one for the player. The starting and ending points for the new path determine if the original path is modified or is replaced.

The different possibilities for route modification are depicted in figure 5.1.1 Route Modification. Suppose we want to modify a path P consisting of points S through E for a player A as shown in figure 5.1.1.a, thus creating a new path P'. If the new path starts over the existing path from a point S', all points from S through S' of the original path are added to the start of the new path. This is shown in figure 5.1.1.b. If the start of the new path is not over an existing point in the original path, for example it starts from the initial location of player A, the original path P is totally discarded as shown in figure 5.1.1.c. If the new path ends over an existing path at point E', all points of the original path from E' to E are added to the end of the new path. This is depicted in figure 5.1.1.d. If the end of the new path is not over an existing path, no more locations are added to the new path at the end. This example is shown in multiple figures in 5.1.1.a, 5.1.1.b and 5.1.1.c. The existing path may consist of a series of routes but the basic principle to route editing remains the same as described here.



5.1.1.a **5.1.1.b**

5.1.1.c **5.1.1.d**

Figure 2. Route Editing - Dotted lines indicate discarded path and full lines indicated new/existing path

5.1.2. Track Rule

The track rule allows the coach to seed in real football play data into the simulator. Such data may be used to study plays that have happened in real time. These data may be used to study opponent behavior, individual player's and its team's strengths and weaknesses, mistakes committed by each player, develop better strategies to obtain the desired results etc. The integration of track rule allows the coach to create realistic scenarios, study past events and prepare better for future scenarios

The simulator reads a file containing locations of the players on the field for the entire duration of the play. The file data is converted to the simulator's field coordinate locations. The offense and defense formations are created from the initial locations of the players on the field. The entire data for a specific player provides a track rule for that player. This track rule constraints the player to be located at specific locations on the field at specific time during animation. Collisions are already accounted for in such real data track rule. Consequently, the collision system is turned off for a player if its rules consist of only track rules and is turned on whenever an active non track rule is performed by that player.

The track rule and the route rule have similar characteristics. Both have an associated path that the player runs on. The route rule is driven by the speed of the player, but the track rule has an associated time with each point on the path. The track rule is created from real play data only and the coach cannot create his own track rule. But the coach may edit a previously created realistic scenario that contains track rules. The coach may alter the path of a player performing a track rule or add other simulator rules from the interface. Once the path for a track rule is modified, the simulator converts it from the point of modification to a route rule. The track rule may only be specified by a series of location and time pairs. Since the new path does not specify any time constraints, it cannot be specified as a track rule.

5.1.3. Parameters Specification and Customization

We have already described the use of parameters specification and customization in the previous section and how it is performed in our simulator. The parameterization and customization are performed in the Sketch to Plan phase.

In the original Interactive Football Playbook, the coach could not differentiate between characteristics of two players like speed, strength or between two rules of the same kind like different route speeds etc. We have identified the various parameters and allow the coach to customize these from the interface. The coach may customize a scenario in the Sketch to Plan phase.

5.2. Sketch to Modify

The coach may use the sketch to modify phase to modify the rules at run time. The original Interactive Football Playbook provided the ability to specify rules only at time t = 0; that is, all the rules need to be mentioned at the start of play [1]. We discuss in this section the need to specify the rules at different times during the animation and how we implement it in our simulator.

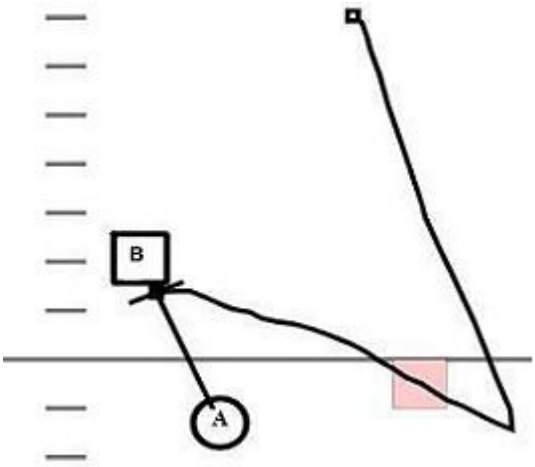
5.2.1. Mid Play Rules

The requirement to specify all the rules at the start of play is tedious and tricky as the scenario becomes more complex and involved. Often, in football, the players improvise in real time and modify their original plan. The coach wants his players to visualize the effects of variations to the original set of rules. The coaches need a technique to simplify the rules specification for the entire play duration. They expressed their ideas to be able to mention rules at different instances of time [1]. This provides the potential for the coach to handle unprepared situations and also improvise.

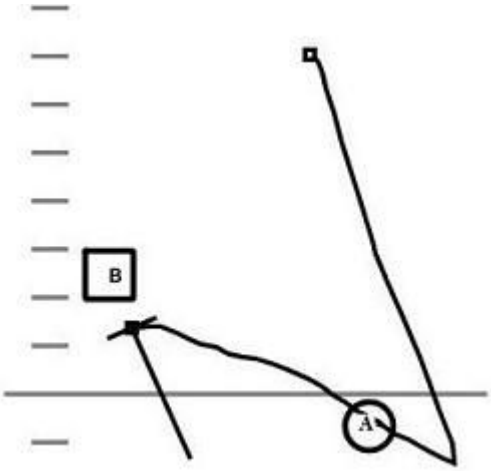
5.2.2. Implementation

The coach specifies the rules for a player and runs the animation. At a specific time instance during animation, he may desire the player motion be altered. He may press the pause button at that time and specify the new desired rule for the player. The new rule drives the future behavior of the player and is chained to the current rule. But the current rule may not have completed at the time of the new rule specification. At the time of the new rule specification, the coach is satisfied with the player behavior up to that time and specifies the rule for its future behavior. We convert this past behavior to a new track rule and associate the original path the player had traversed to the new track rule. This ensures the new player behavior is consistent with the past behavior that the coach is satisfied with. The new rule specified is chained to this track rule.

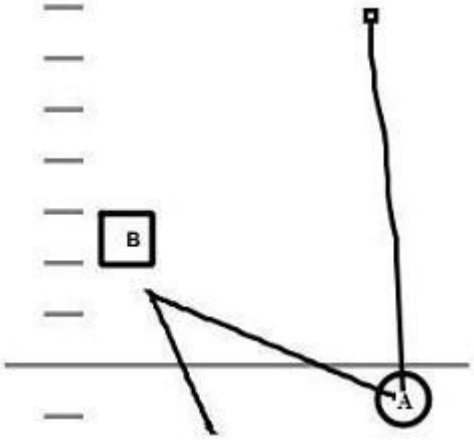
We assume that the player's environment is unchanged before and after the mid play rule specification. If the environment is affecting the concerned player such that the converted track rule is inconsistent with its previous behavior, then the coach may have to specify a new set of rules for that player to obtain the desired behavior. The simulator does not automatically create new rules or change the player behavior if its environment has changed. The following figure, 5.2.2 Mid Play Rules, explains how changing a player environment after mid play rule specification may render the modified player behavior inconsistent. Suppose a player A is blocking a player B for 2 seconds and is then running route as shown in figure 5.2.2.a. The coach, after 3 seconds, modifies the route by specifying a new route for player A. The position of player A at the time the animation was paused is shown in figure 5.2.2.b. The first 3 seconds of player A behavior is converted to a track rule where it was blocking player B for 2 seconds and had run a part of its original route. Figure 5.2.2.c shows this modification along with the new specified route. When the animation is run again with this scenario, player A past behavior is consistent with its original behavior of blocking player B for 2 seconds which is followed by the new route rule. But now if the location of player B changes, for example a new route rule is specified for player B, the track rule for player A may now not be blocking



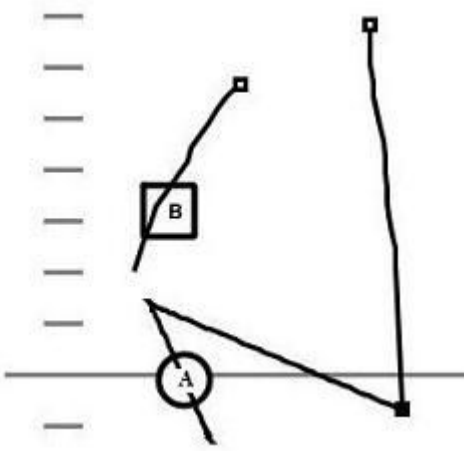
5.2.2.a



5.2.2.b



5.2.2.c



5.2.2.d

Figure 3. Mid Play Rules

player B and is inconsistent with the original behavior. In figure 5.2.2.d, when the player A arrives at player B's original position, player B has already moved to a new position and A is no more blocking B. In such cases, we assume the coach is responsible to change player A behavior whenever he changes its environment.

5.3. Sketch to Correct

The sketch to correct phase of the simulator allows the coach to implement goal based editing. It allows the coach to author behavior by describing what he wants rather than how to achieve it. In many situations, the coach may desire certain behavior for a player, but may find it difficult to specify it or to achieve specific spatial goals at run time. The coach may specify the best approximation of rules and then use the sketch to correct module to correct the player position at run time.

The coach corrects the behavior of a player by specifying spatial and temporal constraints on it. During animation, the coach repositions the player to the desired location on the field. The location serves as the spatial constraint and the simulated time of repositioning provides the temporal constraint. The simulator computes the player behavior by optimization subject to the constraints placed on the player.

5.3.1. Outcome Authoring

The sketch to correct module can also be characterized as outcome authoring. The coach specifies the outcome of the player behavior and the simulator computes the behavior by optimizing its motion. The coach may use a sketch to correct phase to obtain the desired simulation by specifying the outcome of the player's behavior rather than tediously manipulating the player properties and the behavioral parameters.

5.3.2. Implementation

In this section, we discuss the steps the coach performs to specify the constraints and how our simulator optimizes the player motion to satisfy these constraints. The coach may press the pause button at any time during animation and reposition a player. To reposition a player, the coach needs to select the reposition player check box and drag the player to the repositioned location. The simulator then optimizes the parameters associated with the player's rule and satisfies the repositioning constraints. The repositioning constraint attempts to position the player at the repositioned point at the repositioned instance in the simulation. The optimization creates a new path for repositioning the player. We aim to minimize the difference between this new path and the original path for the player until the repositioned time.

We calculate the new path to the repositioned location by making an initial estimate of it and we then iterate to obtain a better approximation. The initial assumption of the new path is a straight line between the player's current location and the repositioned location. The player is currently running the original path and we want it to run this new path so that it may reach the repositioned point. At each instance of time, the player is influenced by these two paths with different velocities. We have used the concept of ease-in-ease-out curves to move the player from the original path to the new path. We optimize the difference between the new and the original paths subject to the parameters of this ease-in-ease-out curve.

The equations for the curve and the objective function are discussed next. The ease-in-ease-out curve has its input and output values clipped between 0.0 and 1.0. An example of an ease-in-ease-out curve is shown below in figure 5.3.2.a. As the time varies from 0.0 to 1.0, the weight or influence of the new path varies from 0.0 to 1.0. This means the influence of the original path decreases from 1.0 to 0.0. The sum of the weights has to be 1.0 at any point in time. The equation of our ease-in-ease-out curve is shown in the equation below.

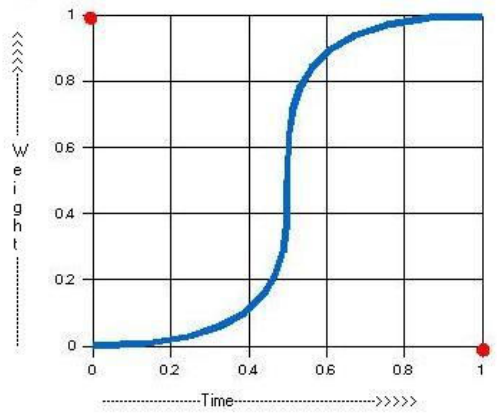


Figure 4. Ease-in-ease-out Curve

$y = ax^2 + bx + c,$

Here,

$0.0 \leq x \leq 1.0$ is the time,

y is the weight of the new path ,

a, b, c are the parameters

These parameters a, b and c are computed to minimize the cost function which is as given below:

$Cost = \sum Distance(x1i, y1i, x2i, y2i) + P,$

Here,

$P1$ is the original path and $P2$ is the new path,

$(x1i,y1i)$ is the i^{th} point of $P1$,

$(x2i,y2i)$ is the i^{th} point of $P2$,

P is the penalty if the player is not located at the repositioned point at the repositioned time

We optimize this objective function using a gradient descent approach by numerically differentiating the function with respect to each parameter. We compute an initial cost by making an estimate of each initial $(x2, y2)$ pairs and the parameters a, b and c. The

derivative of the cost with respect to each parameter provides the gradient and we update the respective parameter value proportional to the negative of this gradient. We compute a new cost for this new set of parameters and iterate the numerical differentiation process to obtain a better approximation of the parameters. We stop iterating when the difference in cost between successive iterations is below a predefined threshold value. The final parameter values optimize the cost and specify the desired track rule for the player.

We have limited our optimization to modifying a subset of parameters namely, track rule formation and leverage rule modification. The future work section discusses a list of various parameters that may be optimized to satisfy the repositioning constraint.

5.3.2.1. Track Rule

In this method the optimization calculates a new track rule to signify the set of rules performed by the repositioned player until the repositioned time. The coach corrects a player's motion by repositioning it at time t to a new location L. The player may have performed any subset of rules until now to traverse a path T1. We convert this path T1 to a track rule R1 and compute a new track rule R2 that satisfies the repositioning constraint. The player resumes its original rules after the track rule R2.

5.3.2.2. Leverage Rule

A leverage rule instructs a player to be on either inside or outside with respect to another player. The leverage parameter specification may be trick, especially if the both the concerned players are constantly moving on either side of the center. Our simulator allows the coach to reposition a player maintaining a leverage during animation. The simulator computes an appropriate new leverage rule using optimization. The leverage parameter is optimized only if the current rule has an associated leverage. The parameters that are adjusted to optimize the objective function are the leverage type and the leverage distance. The optimization works similar to the implementation in the track rule case.

6. Discussion

We provide a basic summary of what we have presented in this paper, its purpose and our approach. We also generalize our approach to various domains and describe how the simulator may be used by various domain experts to fulfill their requirements for simulating strategic dynamic virtual agents in their domains.

6.1. Summary

We intend to build a simulator that allows various domain experts to author strategy for the dynamic virtual agents. The programming logic to implement the strategy is performed by our simulator and the domain experts need not be aware of such tedious details. Much research, therefore, has gone into the development of realistic simulators and integration of real sensor data, so that the domain experts can produce realistic scenarios without scripting all the desired behavior. The simulator uses the space and time constraints specified on an agent to optimize its behavior. The simulator presents to the expert several plausible behavior options, from which the expert may choose one as a final strategy or as a seed to a new strategy.

Our simulator is used to fill in the simulation details to help the experts visualize their strategy. The expert may use our simulator as a teaching tool to author a specific scenario and plan for future events. The expert may seed in real sensor data to study a past event or modify it to realize a better strategy. The simulator allows the expert to program an agent, modify its behavior, specify constraints on it and visualize future possibilities.

6.2. Generalizing to other domain

Although the work performed in this paper is in the American Football domain, the concepts presented can be extended to various other domains. The formations of teams can be interpreted to various domains such as battle simulations, game simulations,

obstacle avoidance etc. The rules specified may be specialized to each of these domains. The cover rule, for example, may be used by an agent to infer the motive of another agent in strategy games, battles etc. The current set of rules may be expanded or modified to cover different requirements for other domains. The addition of track rule has provided a new dimensionality to specify behavior for an agent in terms of its motion that is read from a file.

The naming conventions and the set of rules may change with respect to various domains but the methods for authoring of behavior, specification of constraints, visualization of past and future events etc remain mostly unaltered. An expert from any domain may specify behavior for agents in terms of his own domain in our simulator. We will have to provide the expert with a technique to specify such behavior in our simulator. This means that just the set of rules, grammar and the corresponding actions change, but the specification, visualization and optimization in the core remains the same. We intend to generalize our simulator to various domains and discuss this in further detail in the Future Work section.

7. Future Work

The enhanced Interactive Football Playbook has wide scope to be explored for various possibilities with generalization, learning its decision support method and modification of different parameters. Due to the scope of the current project they have not been implemented and are mentioned as part of future work. We summarize these in this section.

7.1. Different Parameters Modifications

In this paper, we have implemented decision support using track rule and leverage rule formations. We replace the existing set of rules using either track or the leverage rule and modify the parameters for the same to attain the desired behavior. But these two rules and their parameters do not form an exhaustive list. There are various other rules and parameters that may be modified to obtain the desired behavior. These parameters include player speed, rule distance, rule time etc. We have summarized these parameters in the table 7.1 Parameter Modification Table and explained in detail in this section.

The first parameter we consider in this section is player speed. The player speed may be varied to satisfy the repositioning constraint. The player speed parameter is customizable and the coach specifies it from the user interface controllers. The coach replays the simulation and may find that he desires a different speed for the player. By way of trial and error he may get a better value for the player speed parameter. To avoid this tedious method, the coach may choose to reposition a player at run time to the desired location and allow the optimization to calculate the player speed parameter.

	Parameter	Player Speed	Rule Radius	Rule Execution Time	Route Replacement
Original Rule					
Route		Forward or behind in the same path			After rule execution close to completion
Pursue		Forward or behind in the same path			
Block		Forward or behind in the same path		Forward or behind with respect to the child rule	
Cover		Forward or behind in the same path	Closer to or away from the target player		
Avoid			Closer to or away from the target player		
Wait				Forward or behind with respect to the child rule	
Track					Any player performing track rule

Table 3. Parameter Modification Table

The player speed can be changed to satisfy the repositioning constraint only if the player is repositioned in the same path, either forward or behind the current position. If the repositioned location is different, a new path will be required for the repositioning satisfaction. Hence a different parameter or a new rule will be required, merely changing the speed parameter may not help. The path means different for various rules. For route, the path is the same path that the player runs. For cover and pursue, the path is at each instance the line between the covering or pursuing player and the target player. For block, the path is a combination of the blocking boundary angle, leverage and line between the blocking and the target player.

The following figures show an example scenario where a player A running a route is repositioned ahead of its current location and on the same path. Player B is pursuing player A. The coach has specified a speed for the player A but also wants it to stay ahead of the pursuing player B. The player A speed has to be increased for this. At the given instance, the coach decides to reposition player A. The coach desires player A be repositioned as in figure b below. The optimization then calculates a new increased speed for the player A and displays it as one of the options for the coach to choose from.

The second parameter is the distance for the cover and the avoid rules. This is the distance the covering or the avoiding player must maintain with respect to the target player. Suppose the coach repositions the covering or avoiding player A closer to or away from the target player B. This behavior can be obtained by updating the distance parameter of the corresponding rule. This gives the coach another option along with the existing set of options to choose from. The coach may play all these options and select the one that best matches his needs.

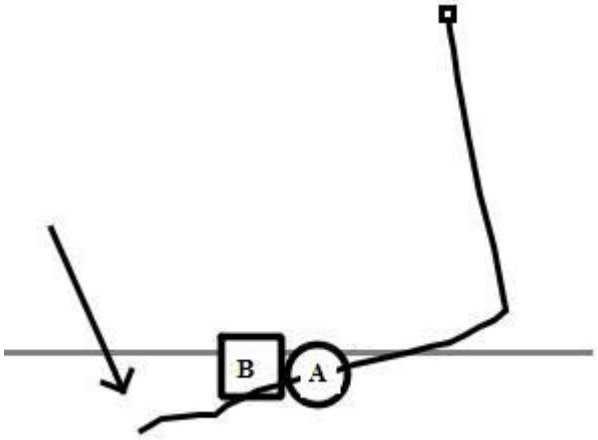


Figure 5. Speed Modification Original Position

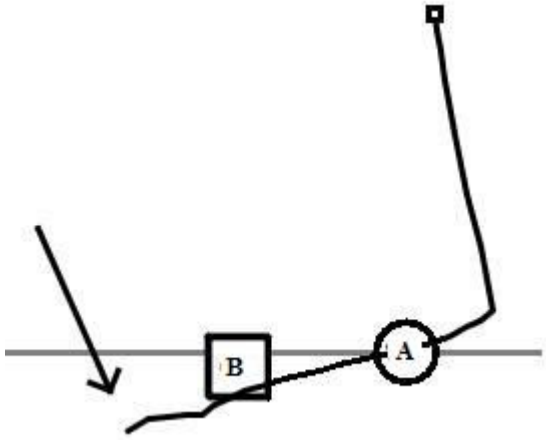


Figure 6. Speed Modification Repositioned Position

The wait and block rules have an associated time parameter. These rules can be chained and any set of rules may follow it. Suppose that a player has a route rule that follows a wait rule. The coach decides to reposition the player when it is running the route. The repositioned point is close to the current path and behind the current position. This example is similar to the above case where the player needs to run with lower speed. But this could also mean that the player needs to wait for a longer period in the same position and then run with the same speed. Thus the repositioning constraint is not only bound to the current rule but also to the parent rule. The parent rule consideration with respect to time is restricted only to scenarios where block and wait are the parents.

The route replacement is the next parameter we discuss in this section. It is used in situations where the coach repositions either a track rule or a route rule that is close to completing its execution. Let's say the coach wants to reposition a player performing a track rule. This may be to modify the output of an optimization or to change player rules from a real play. The original track rule is divided into two parts; part one before the repositioned point and part two after the repositioned point. The part one is used as a new track rule. The part two now needs to be converted to a route rule since we do not have any information for a track rule starting from the repositioned point. A new track rule can be added to the scenario only if the various requirements for optimization are satisfied or using real sensor data. This part was described earlier in the Sketch to Correct section using Track Rule Formation. The route rule replacement is also useful if a player running a route rule has completed most part of its execution. The coach specifies the route using free hand path drawn on the simulator and this may cause slight errors to slip in. The repositioning close to completion suggests that the coach desires the route rule must exist only until the point of reposition. The original route rule is truncated and the existing set of child rules are then added after the new route rule.

7.2. Generalize Scenarios and Create Repository

We envision a general framework to be used by various domain experts, not just football coaches. We need to identify the various target audience, their requirements from the simulator and their usage pattern. The simulator must be made generic to specify scenarios, rules, grammar and constraints in common terminology or create various specialized options for each domain.

Currently, the coach has to create formations and then create a scenario by using two formations and specifying various rules on the players. The next step is to customize different parameters for the rules and the players. The entire scenario specification by the coach may become time consuming if he wants to run and store various simulations with minor changes from one another that explore the different outcomes. The repetitive work can be minimized if there is a technique to create a repository of the scenarios specified by the coach. By generalizing the scenario, the associated rules, parameters and formations, we are saving time on part of the coach and also trying to better model the coach's decisions.

7.3. Learning Choice

In optimization, we present the coach with a list of options to choose from that match his desired behavior. This list is not ordered from the best match to worst match. It is generated by changing different parameters and is random in terms of which parameter we choose to modify first. Once the coach selects a particular choice from the list, there is no concept of recalling or learning his choice. If later, the coach specifies a similar constraint to expect similar desired behavior, a similar list is output instead of automatically generating his preferred choice. The learning of his choice can help us reduce unnecessary options calculation and the repetition of the coach's selection process. The last choice of the coach may be selected and replayed for consideration by the coach. If the coach rejects it, then we may go for the various different options. If we

can represent the set of inputs and outputs in an optimized manner within the simulator, the playback of the coach's previous choice may be performed easily. The inputs consist of the scenario and the repositioning constraint. The scenario is specified by the relative positions of the different players in the neighborhood of the repositioned player, while the repositioning constraint is the relative location and time of the constraint specification. We also need to check if the learning of choice would be any beneficial to the output list generation or it would cause more time and processing delay.

7.4. User Studies

The coaches' feedback in the original version helped us create a definitive plan of action. The user interface support and mid play rules were mainly inspired from the feedback. The ideas helped us envision the scope for this current project. User Studies is an important part of this project also. We need to identify how useful the decision support and the listing of options are for the coaches. Questions like how easy is it to use and understand the interface, what changes are required for the simulator to be more realistic, what changes are required to the current method of specifying constraints and choosing options, does the given set of options match the expected behavior, is there any set of unusual behavior that needs to be rectified etc remain to be answered. The user studies concerning different domains and a generalized scheme that optimizes the needs of all the experts are another major work that needs special attention and could not be completed in the current scope.

8. Bibliography

[1] Christoph Neumann, Jonathan Dodge and Ronald A. Metoyer. Interactive Football Playbook. Master's thesis, Oregon State University, July 2006.

[2] Craig W. Reynolds. Steering Behavior for Autonomous Characters. In *Proc. of Game Developers Conference*, pages 763–782. Miller Freeman Game Group, San Francisco, CA, 1999.

[3] The Physics of an Elastic Collision http://director-online.com/buildArticle.php?id=532

[4] Christoph Neumann. End-user programming of 3D virtual agents. In *IEEE Symposium on Visual Languages and Human Centric Computing*, pages 285-286, Sept 26-29 2004

[5] Deep Green http://www.darpa.mil/ipto/Programs/dg/dg_approach.asp

[6] Christoph Neumann, Ronald Metoyer, Margaret Burnett. End-user strategy programming. In *Journal of Visual Languages and Computing*

[7] Jon Kleinberg, Eva Tardos. Algorithm Design book, pages 261-266.

[8] Ronald A. Metoyer and Jessica K. Hodgins. Animating athletic motion planning by example. In *Proc. of Graphics Interface*, pages 61-68, May 15-17 2000.

[9] Paul R. Cohen, James A. Davis and John L. Warwick. Dynamic Visualization of Battle Simulations. Massachusetts Univ Amherst Dept Of Computer Science, 2000

[10] Ronan Boulic, Nadia Magnenat-Thalmann, Daniel Thalmann. Coach-Trainee: A New Methodology for the Correction of Predefined Motions. In *Proc. of Eurographics Workshop on Animation and Simulation*, pages 1-14, 1990.

[11] How the physics of football works http://entertainment.howstuffworks.com/physics-of-football3.htm