Game Designer Behaviour Modelling Interface

Dr John R Rankin and Guy A Suermondt
Computer Science, La Trobe University

{gasuermo,johnr}@cs.latrobe.edu.au

ABSTRACT

A new Behaviour Model for modelling NPCs has been developed, which is ethologically based and generates context sensitive behaviour. The model has been successfully used to simulate realistic NPCs with essential complex game behaviours such as cooperative and herding behaviours. However setting up creature models requires the user to design top-level factories and some parts of them are customised by XML input files and the rest is hard coded. To avoid a game designer having to have a deep knowledge of the model, an interfacing program has been designed that will generate the XML files in the required format. This paper outlines the design of this interfacing software. As a result it is now easy for game designers to create and test NPC behaviour models outside of the game.

1- INTRODUCTION

It is an important issue in games today to have more believable and complex behaviours in game NPCs [3]. This requires good behaviour modelling software with the properties of fast decision making and capable of handling realistic complex behaviour without using a large amount of memory. Such a model has been recently developed by the author [5].

The model adapts two complementary models those of Blumberg and Tu and is based on ethological concepts, which partly strengthen the influence of context on the action decision algorithm.

On one hand, Blumberg's model [1] is grounded in a rich ethological abstraction layer essentially composed of Internal Variables and Releasing Mechanisms and Group Behaviours, which are sets of Behaviours. Inner Variables, which represent a character's internal stimuli can either be triggered by external events or behaviour values. Releasing mechanisms analyse external events and return values correlated to the behaviour relevance for being expressed. Between Inner Variables and Releasing Mechanisms, there are Behaviours bringing together Releasing Mechanisms and Inner Variables. Behaviours compete and the character's next action is derived from the winning Consummatory Behaviour [1].

This strong abstraction layer is the fundamental strength of Blumberg's model. Thus, it may facilitate the game designer implementation of NPCs by providing game designers with elementary blocks in order for them to build their NPCs. However the relation between Behaviours and Inner Variables is not flexible and simple enough for NPC modelling: different notions of internal stimuli cannot be modelled and changing the weight of a specific behaviour may require the designer to change its inhibitory power as well as its influence on the related Inner Variables [1].

On the other hand, Tu's model [4] stresses the mutual influence of action and perception, conveyed by the notion of Desire. A Desire, which is the normalised expression of an Inner Variable, directly influences the character's direction choice. In doing so the character perceives objects that are implicitly correlated to its Desires. At the same time, Inner Variable values are updated in regard to the new external situation.

Here the choice of the behaviour is done through a tree structure where nodes are behaviours and branches conditions leading to these behaviours [4].

The relationship between the character's Desire and the character's choices is the key point of Tu's model for game character modelling. Its disadvantage is the static tree structure that does not allow a designer to easily change the hierarchy among the character's Behaviours.

The new Behaviour Model merges the strong conceptual abstraction layer of Blumberg's model with the action-perception mutual interaction concept from Tu's model. Thus Releasing Mechanisms of this model return a

value that is a trade off between the relevance of the related Behaviour object for being expressed and the attractiveness of the external scene in regard to a character's Desires. In addition, the new model introduces a new way of modelling Inner Variables. An Inner Variable is a combination of Goals and Motivations. A Goal is an internal stimulus triggered by an Essential Variable, which is a character homeostatic variable. Thus a Goal can be considered as reflecting the influence of a long-term objective. Motivation is an internal stimulus triggered by the occurrence of a specific external or internal event. As a consequence Motivations can be considered as the character's shorts-term objectives. Finally the last feature is to tie a Strategy object to a Behaviour object. A Strategy object is a plan to achieve the implicit Behaviour Desires and is used by related Releasing Mechanism objects to assess the relevance of the Behaviour for being expressed [5].

The new Behaviour Model still requires a deep technical knowledge in order to create specific NPC behaviours in regard to the game world. To create new specific behaviours, designers can either customise the input XML files or add extra behavioural factories in charge of building new Inner Variable, Goal, Motivation, Releasing Mechanism, Strategy or Behaviour objects [5]. In order to facilitate the XML file customisation an interface program can be created. A designed solution of such an interfacing application is presented in this paper.

The following section presents a typical situation of a NPC that a designer needs to implement.

2- BEHAVIOURAL MODEL FOR DAVID

David is a NPC required in an interplanetary travel game [5]. He is programmed to help the player who travels from planet to planet solving problems and yet he has fallibilities as well. For example his attention may become distracted and he may become intoxicated under certain contexts.

A simple behavioural model for the NPC David is presented below. (See Figure 1.)

David's Model is built in two steps: first his internal mechanism is built and then his behavioural responses set is designed.

In order to build David's internal mechanism, David's Goals, Inner Variables and Motivation Releasing Mechanism need to be designed.

Three main goals drive David's life: surviving 'Survive_G', helping the user 'HelpUser_G' and satisfying some bad pleasures 'Vice_G'. The purpose of this last personality trait is to make David more realistic and to make the game more entertaining. Due to this personality David may perform some unexpected actions, which may disturb player's plans or just amuse the player.

From these three goals a set of essential variables, Survive_ES, Vice_ES, HelpUser_ES, are created as well as a set of Inner Variables from which are derived David's Desires that influence his Releasing Mechanisms and Behaviours.

Here the essential variables model the game physics underlying the David's Goals. For instance Vice_ES can be an increasing function that decreases only when David successfully satisfies one of his vices. Thus the more resisting David is to his vices the stronger the goal is which guarantee that David will sometimes be tempted. Survive_ES can be an estimator of David's wealth. Therefore the less money David has, the more easily it will become the player's friend. Finally HelpUser_ES is an estimator of the player kindness for David. The more kind the player is with David the higher the estimator is. Thus David will not be forced to help the player if the player behaves badly toward himself.

Connecting Inner Variables with Goals is done in regard to how David's goals can affect his Desires. In these case the link is 1-1.

Finally the last step to build David's internal mechanism is to design his Motivational Releasing Mechanisms.

In this scenario Motivational Releasing Mechanisms are designed so that first David helps the player regardless of his estimation of the player's kindness and secondly David is tempted when an unexpected vice situation occurs.

After creating David's internal mechanism, his behavioural response set should also be created. Here David is expected to behave in three particular ways. First David should share his knowledge with the player, then he should try to become the player friend and finally David likes drinking and seducing women.

Thus David Strategies, which are pre-planned to achieve his goal, can be designed, Str_Drink, Str_SeduceWoman, Str_GetEmployed, Str_Suggestion, as well as David's Releasing Mechanisms, which are the typical situations where David can express his Strategies.

Once this step is done behaviours are created so that David's Desires and David's Behavioural responses are brought together.

In Figure 1, the behavioural branch that wraps the Stratey Str_Fight is drawn in order to illustrate a behaviour family containing two Appetitive Behaviours.

The Behavioural Model structure for David is listed below.

Goals:
VICE_G: reflecting how strong the character's personality trait [2] related to vice is.
SURVIVE_G: reflecting how strong the character's personality trait related to its own survival is.
HELPUSER_G: reflecting how strong the character's personality trait related to player's survival is.

MRMs:
EXCEPTIONALVICESITUATION: the MRM triggers when an exception situation occurs such as the presence of an incredibly beautiful woman.
USERINDANGER_MRM: the MRM triggers when the player is in extreme danger.

Inner Variables:
VICE_IV: reflecting how tempted the character is.
SURVIVE_IV: reflecting how important survival issues are for the character.
HELPUSER_IV: reflecting how keen the character is on helping the player.

Strategies:
STR_DRINK: take the drink and then drink it.
STR_SEDUCEWOMAN: discuss, make her laugh and kiss her.
STR_GETEMPLOYED: offer services, negotiate salary.
STR_SUGGESTION: say an information to the player.
STR_FIGHT: fight for the player

Behaviours:
Three Families: Fvice, Fsurvive, FhelpUser

Releasing Mechanisms:
SERVEDRINK_RM: detect a full glass.
ASKDRINK_RM: detect a person that can serve a drink.
DETECTPOSSIBLETARGET_RM: detect challenging woman.
POTENTIALBOSS_RM: detect a person that can hire the character.
POSSIBILITY_RM: deduce meaningful information from the character's database.
OPPONENT_RM: detect an opponent to the player.

3- SOFTWARE REQUIREMENTS:

From the details of the sample game character Behaviour Model Structure of the previous section we can see what typical requirements the interfacing application needs. The interfacing application must firstly ask the user for the new creature's name and have its species selected from a drop down list. Then the application must provide the game designer with a window containing access to entry pages for all the entities listed in the structure above: a page for editing NPC goals, a page for adjusting Internal Variables, and pages for adjusting RMs and MRMs. When the game designer selects the desired species then the defaults for the corresponding behavior factory are loaded into the text boxes on the data entry pages. These defaults show the creature's Goals and the game designer is then able to make changes to the Goal impulse function such as the choice of function and function parameters. These species defaults also show the consummatory behaviours and the RMs connected to them. The game designer is then able to make changes to the RM internal structures such as the choice of function and constants in the RM triggering equations. Likewise the defaults show the Inner Variables and the MRMs connected to them. The game designer is then able to make changes to the MRM internal structures such as the choice of function and constants in the MRM triggering equations. Similary, the game designer can for each sensor on the creature modify the detection probability curves by choice of function and constants in the equation. Once these choices have been made the user terminates the interfacing program, which will cause the program to save the behaviour model details to appropriate XML files.

Having made these adjustments the game designer can create a number of creatures of the same species with a wide variety of performance differences in behaviour. However the choices of parameters and functions made do not give the game designer any clear idea of how the new creature will perform in the various situations it will face in the game. Because of this we provide another stand alone application that will show creature behaviours outside of the game software. The model tester application inputs the behaviour model details for a specific creature from its XML files and then displays the creature in an empty window. The user can then set a terrain type (the window is a single game background cell) and add other creatures of any species to see what behavioural responses the creature makes. The additional creatures are static and no modelling of their behaviours is made. The model tester application therefore provides minimal testing of the possible responses and behavours of the new creature before that creature is incorporated into a large game software.

4- CONCLUSION

Requirements for a Behaviour Model interface application have been established which clearly show the feasibility of the design and therefore we are able to provide such an interface to game designers that will enable them to enter the necessary data for constructing the behaviour model for all creatures in the game without the game designer having to understand the internal workings of the new Behavioural Model. Game designers however still need to understand the ethological theory underlying the design of creatures to achieve optimal results with the new Behavoural Model. The interfacing application does not allow the game designer to change the hard-coded classes of the Behaviour Model factories. Instead we provide parameterised factories for Goals, RM, MRM, IV, and the sensor detection selection algorithms. Data for these parameterised objects including choices of functions and the constants involved in them are read from an XML data file.

As a result of this research, creatures do not have to be recreated from scratch: modifications can be made to default creature parameters. Complex creature behaviour modelling is now achievable, and a simulator for testing behaviour models before they are incorporated into games has been described.

In future work we hope to extend the software to increase the complexity of game creatures by further additions to the interfacing software. One way is to allow game designers to create generic behaviour based on richer physics (more functions and parameters to choose from) and adjustable parameters inside the behaviour algorithm. Another way is to increase the amount of environmental factors influencing creature behaviour. We also intend to extend the model testing application to allow multiple dynamic creatures for testing creature herding and cooperation.

5- REFERENCE SECTION

[1]. Blumberg.BM (1997), *Old Tricks New Dogs: Ethology and Interactive Creatures*, PhD dissertation, Massachusetts of Technology USA.

[2]. Icek Ajzen (1988), *Attitude, Personality and Behavior*, Open University Press, Milton Keynes, UK, pp. 1-8, pp. 21-24, pp 63-64, pp 89-93, pp. 109-113, pp. 143-150

[3]. Morris.D &Rollings.A (2000), *Game Architecture and Design*, The Coriolis group, Arizona, US

[4]. Tu.X (1999), *Artificial Animal for Computer Animation: Biomechanics, Locomotion, Perception, and Behavior*, Springer-Verlag, Berlin, Heidelberg, New-York.

[5]. Suermondt.GA, *Ethologically Oriented Context-Dependent Behaviour For Non Player Characters*, thesis submitted for MSc degree, La Trobe University, December 2003.

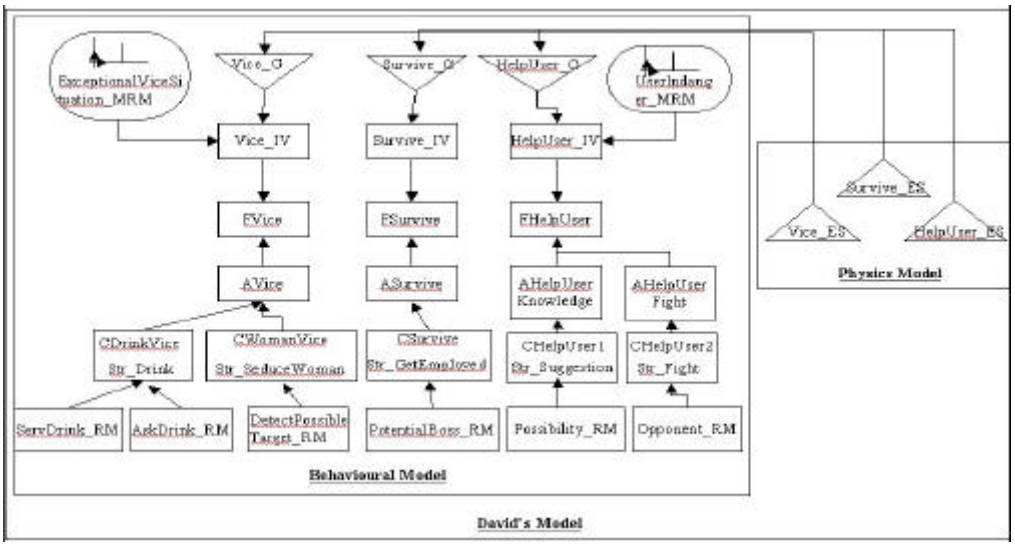

Figure 1. David's Model