Verification of FSM using Attributes Definition of NPCs Models

***Chong-Han Kim*†, *Seung-Moon Jeong*†, *Gi-Taek Hur*††, *Byung-Gi Kim*†††

†Dong Shin University Digital Contents Cooperation Research Center,
††Department of Digital Contents, Dong Shin University,
†††Department of Computer Science, Chonnam National University

Summary
The NPC (Non Playable Character) model is a very significant factor in the area of the on-line computer games and the design of virtual space systems. FSM (Finite State Machine) is the most widely used algorithm which uses the finite states to represent the behaviors of NPCs. The correct specification about the artificially intelligent NPC model prevents us from losing the resources in the phase of implementation, and makes it possible to verify the suitability for the requirement specification.

In this paper, we defined the property about the behavior pattern of a fish object when we construct the virtual ocean, and we formalized and established the interrelationship of behavior pattern by environment changing.

We designed and implemented NPC fish models which have a formal property in the virtual ocean by using FSM algorithm. And we proved propriety about the designed algorithm through verifying the functions of the NPC model with SMV, a model checker based on the CTL.

.

Key words: *NPC, FSM, Model Checking, Virtual Ocean.*

Introduction

The concern about the virtual reality has grown and computer users who want a virtual space have been increasing. Virtual space is reconstructed the real world by using computer graphics.
In constructing the virtual space, the reality of the virtual space depends on graphical factors as well as NPC models using AI algorithm [1] [2]. Intelligent NPCs cause computer users to be interested in virtual space. Such a fact is the occasion for the computer users to want more realistic and more intelligent characters [3].
In the constructing the virtual ocean system, AI algorithm of dynamic objects like Fishes is very significant. We define the respective characters and behavior patterns of various kinds of undersea animals. Because the objects are designed based on the definition above, the situation such as food chain, propagation, behavior pattern, and evasion in underwater environments can be expressed realistically.
There are several AI algorithms for designing the NPC model; FSM[4] algorithm using finite state machine, Training algorithm using evolution through training, Gene and neural network algorithm, Flocking method for expression of mob behavior, A* algorithm used in path finding.
FSM algorithm is the most widely used AI processing method and it makes the design of sequential logic or control function of any technique accurate and simple. However, because there is no restriction in the number of the state, it is difficult and complicated to arrange a state diagram in case of requesting a large number of state expressions, and the outside stimulus processing becomes complex. To prevent this situation, we specify the feature to the temporal logic with which the model must satisfy in finite state system, and verify the correctness of models using SMV; the tool of the model checking which decide an existence in either yes or no of the error. [5][6][7].
In this paper, we define the behavior pattern or feature of the ocean floor fishes, and design and implement NPC models with FSM algorithm.
After that, we verify the correctness of the designed algorithm using SMV [8].
In chapter 2, we explain the NPC, FSM, AI algorithm, and verification of the SMV tool for verifying these. In chapter 3, we formally defined the feature of fishes like behavior patterns, the ecology of fishes, and design using FSM algorithms and verify these. Finally we explain the conclusion and hereafter research direction.

2. Relation research

Today, research about the virtual space, PC (Playable Character) and NPC model have been achieved actively around an online game, and effort for the implementation of more intelligent NPC model have been continued. Unlike PC which the user controls directly, NPC model is an AI model which is controlled directly by the computer. It responds to any given event or produces specific situation in any condition in virtual space.
To produce variable situations, we design the NPC models using FSM algorithm which have relatively simple structure; however, if the condition becomes more

Manuscript received July 5, 2006.
Manuscript revised July 25, 2006.

complicated and there are many establishment situations, we need more time and effort to design the model as well as modify or improve the designed models. In addition, we have to endure the loss of the resources caused by wrong design. This brings forward the necessity of verification of design.

2.1 NPC (Non-Playable Character)

NPC model means dynamic objects which are not subordinate at any control of their user and decide by themselves and operate in virtual space. For example, it plays a role as an enemy to the PC (Playable Character) or as a supplementary partner which leads the game smoothly in online-games. Fundamentally, the behavior of NPC is based on the state-transition information and the control structure is defined by the developer. This is an important element which the developers should consider when they draw a plan. Generally, NPC repeats the sense, the thought, and the decision cycle. Fig 1 and Table 1 show the activity of each step.

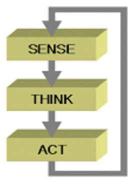


Fig 1: NPC cycle

Table1: NPC activity of each step

Step	***Activity***
Sense	Grasp the situation through sensing the information to be happened in current location from server.
Think	Decide the action which coincided with its own role in the situation.
Act	Order servers that do the deed to be selected.

2.2 FSM (Finite State Machine)

The idea behind the FSM is that a system such as a machine with electronic controls can only be in a limited number of states.
State is the condition to induce any circumstances. Namely, state is the condition value which the system currently has. FSM is an expression of transition which reacts on any affair in the axis of time about a unity object. Transition is change of state. For example, if a switch of light is one of state machines, the state of the switch can be expressed as two kinds, “ON,” “OFF.” This state changes by input condition of “UP,””DOWN.”

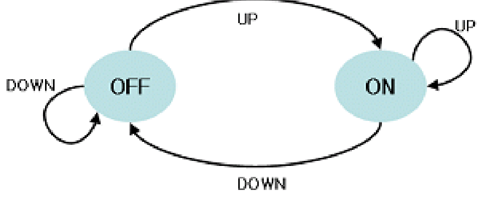


Fig 2: State transition example

The FSM access to the routine using If() or Switch() sentence. Therefore, in the case of the algorithm which reacts identically according to the determined condition, the user can forecast the NPC behavior and it makes them tedious. Also, it becomes the factor frequently evolved and enhanced. There is a FuSM (Fuzzy State Machine) using the Fuzzy theory to improve such defects.
FuSM applies a Fuzzy function with input/output of state, subdivide the input into analog value not digital value like ‘0’, ‘1’. Also, through adding random function, it can get the transition to different state on identical input. In case of FSM, it does only two kinds of operation, “YES” or “NO,” ;on the other hand, in the case of FuSM, it is possible to express various kinds, such as “Definitely YES”, “Generally YES”, “Generally NO”, “Absolutely NO.” It makes the users difficult to predict the reaction of NPC because of the reaction of NPC’s random behavior.

2.3 SMV (Symbolic Model Verification)

The SMV is a model verification tool which distinguishes the fault or truth of the given logic using BDD (Binary Decision Diagram) and temporal logic, so called CTL. Fig 3 shows the structure of SMV.

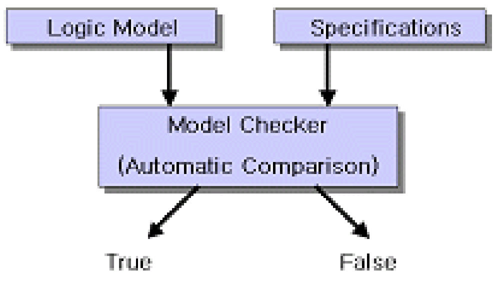


Fig 3: SMV structure

Logic model is the part to express the system through the State Variable, Assign, and Define. Specification model is the part that input the query about the correctness of the model. The SMV Model Checker distinguishes between True and False by checking all the state of system represented about the query.

3. Design the FSM and implement the NPC of oceanic object

3.1 Define the behavior pattern and design FSM of oceanic object

In the virtual ocean, we defined the feature or behavior pattern about each objects to design the NPC model like ocean life. We pursue to convert to the FSM through the standardization of the defined data.
Though there are various species in the ocean, we limit to the feature of fishes in this paper. We can try to classify special character of fish; the food chains, growth condition, inhabitation environment, distinguish of sex, and classify activities of fish; propagation, evasions. Table 2 represents the feature each fish must have. In this paper, we refer to six kinds of the fishes to be marked in a Table2.

Table2: Define the attribute of object

Species (A)	Grade	A1 : Flatfish, A2 : Gurnard A3 : Shark, A6 : Saury A5 : Bream, A6 : Ray
	note	- Define the species name
Food Chain (B)	Grade	B1 ~ B10 grade
	note	- each object can prey upon other object of 3 grade downside self rather it self - B1 is the bottom grade object - B10 is the top grade object - In the case of same species, the relations of food chain dose not form. - It can prey on the other object when the growth condition is 5 grade and higher.
Growth Condition (C)	Grade	C1 ~ C20 Grade
	note	Each grade 6 month difference
Inhabitation environment (D)	Grade	D1 ~ D5
	note	D1 : bottom of sea D5 : surface of sea
Sex (E)	note	F1 : Male F2 : Female
Propagation (F)	note	- Propagation is possible for same species. - Propagation is possible for over the C5
Activity (G)	Grade	G1 ~ G5
	note	G1 : very slow movement G5 : very fast movement

In the Table 2, A~E show the attributes that fishes have, and F, G show the activities. In the characteristics of our system, we only consider evasion condition from the enemy except the condition of predator. For example, sharks do not have any action for preying on lower grade objects, and lower grade objects act like evasion to higher grade objects like shark. We define the behavior pattern of objects which will be inserted in the virtual ocean in Table3.

Table3: Endow the attribute of fish

Species	***Property***	***Activity***
object1	A1, B3, C5, D1, E1, G2	F, G1~G3
object2	A1, B3, C2, D1, E2, G2	F, G1~G3
object3	A3, B9, C10, D2~D4, E1, G2	F, G1~G4
object4	A5, B3, C6, D2, D3, E1, G2	F, G1~G4
object5	A4, B2, C2, D3, D4, E2, G2	F, G1~G5
object6	A6, B6, C9, D1, D2, E2, G2	F, G1~G3
object7	A1, B3, C8, D1, E2, G2	F, G1~G3

We give an attribute value such as the value on Table 3 to each and every object, and induce behavior patterns which satisfy with these conditions. Consider the circumstances that input some condition which different objects exists within a radius of action of object1. About object1 and object2, because they are same species, they do not form the food chain relation, and because the growth condition is not matched, propagation is impossible. But about object1 and object7, because the growth condition coincides with each other, propagation is possible.
If Object3 exists within a radius of action of object1, Object1 move fast fromG1 to G3, because the condition of food chain relation and growth condition are satisfied. About object6 and object3, all condition is satisfied except the inhabitation environment, so they do not form the food chain relation.

3.2 Design and Implement of NPC modeler using the FSM

Table4 is a simple example that each activity is formalized as the determined attribute in advance. In case of propagation, if they are same species and their grade is 5th grade and higher, their movement move dull and propagate by 10% probability. In case of activity, when different species was recognized in the radius of action, G has the maximum value if the inhabitation environment is the same, the gap of the food chain is more than 3 grades and the gap of growth condition is more than 5 grades.

Table4: activity pattern pseudo code

propagation	***Activity***

If(Same Species == true {
else if(C≥5))
{propa = Rand()%10
if (propa = 1)
F="yes",;
"G = Minimum Value";}}
else {no state transition}
	If(Same Species == true)
	{{{
	else if(Same D ==true)}
	else if(Gap B ≥ 3)}
	else if(Gap C ≥ 5),
	G = Maximum Value}
	else{ no state transition}

Fig4 and Fig5 are FSM flowchart to express the activity about the condition up there to be suggested. In case of the propagation, normal state is the stable situation which any condition is not input.

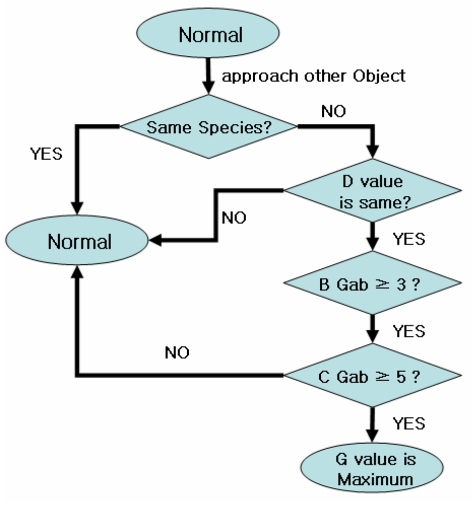


Fig4: Propagation Flowchart

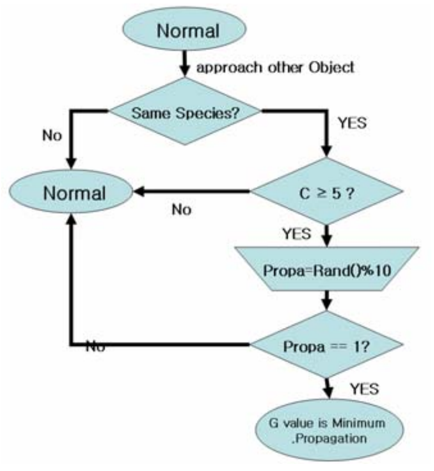
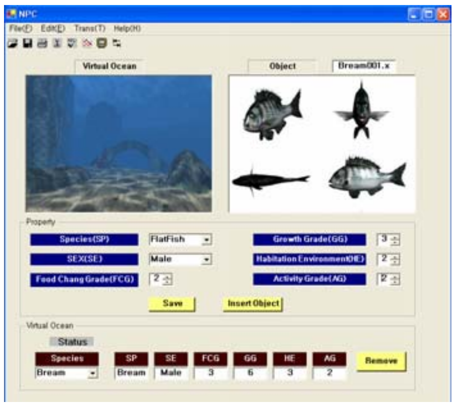


Fig5: Activity Flowchart by Food chain Relation

Fig6 is the interface that input the characteristic of NPC models. Left upper portion expresses the virtual ocean and Right upper portion expresses the 3D modeling data of ocean fishes. In the property window part, the attribute; Species, Sex, Food chain relation, etc can be input. In the status window part, it shows the circumstance of objects inserted in the virtual ocean currently. The movement of the ocean fishes depends on the given attribute value to them.



Bream001.x

Property

Species(SP)	FlatFish	Growth Grade(GG)	3
SEX(SE)	Male	Habitation Environment(HE)	2
Food Chang Grade(FCG)	2	Activity Grade(AG)	2

Virtual Ocean

Status

Species	SP	SE	FCG	GG	HE	AG
Bream	Bream	Male	3	6	3	2

Fig6: NPC modeler interface

Fig7 shows NPC objects acting according to the input of the attribute value. A ray move only in the ocean floor, never float up to the surface of the sea.

Fig7: Virtual ocean applying NPC model

3.3 Method of verification for designed FSM

Verification of designed FSM is achieved through SMV. If some condition is input, one state is changed to other state. Wrong initial design can cause the state not supposed to happen. It leads to serious errors in the system. Fig8 is initial design of the state of a certain specific object using the Active HDL.

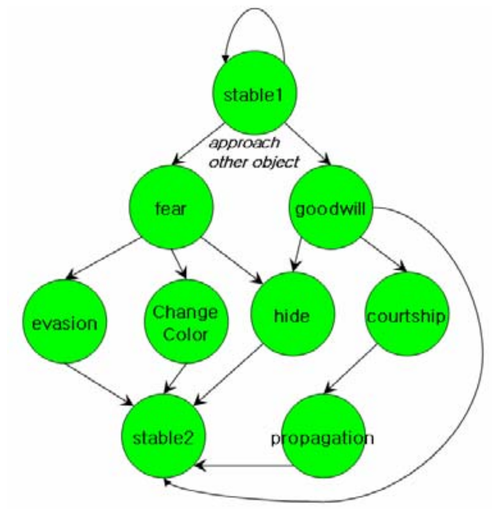


Fig8: FSM of a specific object

Table5 is to convert FSM to SMV code.

Table5. SMV code which expresses states of the object

```
MODULE main
 VAR
   State : {Stable, Fear, Goodwill, Evasion, Changecolor,
Hide, Courtship, Propagation, Stable1};
 ASSIGN
       init(state) := Stable;
       next(state) := case
       state = Stable : {Fear, Goodwill, Stable};
       state = Fear : {Evasion, Changcolor, Hide};
       state = Goodwill : {Hide, Courtship, Stable1};
       state = Evasion : stable1;
       state = Changcolor : stable1;
       state = Hide : stable1;
       state = Courtship : Propagation;
       state = Propagation : stable1;
       1 : state;
 esac;
```

We can confirm that there is no error in the initial design through some setting.

Setting up	***Expected result***
① The object can hide itself if it feels the fear	True
② The object must propagate in a next state if it stays at courtship state.	True
③ The object hides itself if it has the goodwill.	False
④ The object can not do any reaction although it has the goodwill.	True

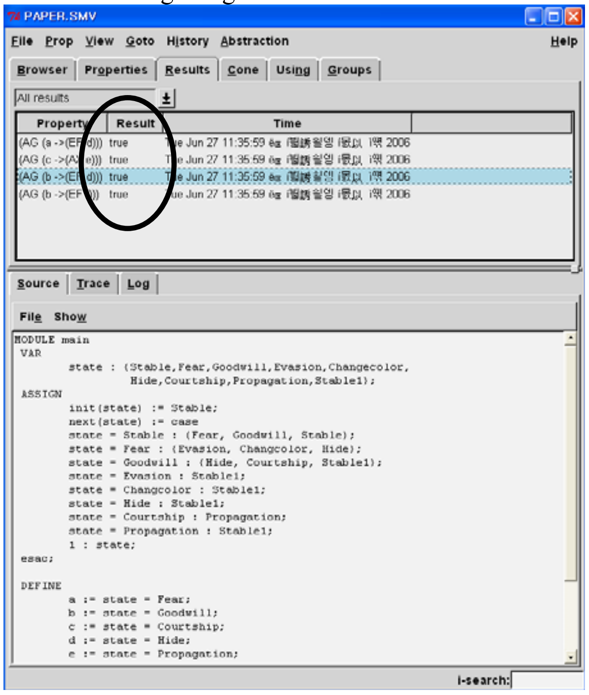
The Table6 is to convert an upside condition to SMV code.

Table6: SMV code according to the setting up

```
 DEFINE
        a := state = Fear;
        b := state = Goodwill;
        c := state = Courtship;
        d := state = Hide;
        e := state = Propagation;
        f := state = Stable1;

  SPEC AG(a -> (EF d))
  SPEC AG(c -> (AX e))
  SPEC AG(b -> (EF d))
  SPEC AG(b -> (EF f))
```

Fig9 is the result after verified the setting up
The circle expresses the result value in this fig.
In the case of the setting up ③ above, the verification result came out to be true even though the expected result should come out to be false. This says that there was an error in the design. Therefore, it needs to be reviewed ecause it is wrong design.


PAPER.SMV
File Prop View Goto History Abstraction Help
Browser Properties Results Cone Using Groups
All results
Source Trace Log
File Show
i-search:

Property	Result	Time
(AG (a ->(EF d)))	true	Tue Jun 27 11:35:59 [illegible] 2006
(AG (c ->(AX e)))	true	Tue Jun 27 11:35:59 [illegible] 2006
(AG (b ->(EF d)))	true	Tue Jun 27 11:35:59 [illegible] 2006
(AG (b ->(EF f)))	true	Tue Jun 27 11:35:59 [illegible] 2006

```
MODULE main
 VAR
        state : (Stable,Fear,Goodwill,Evasion,Changecolor,
                 Hide,Courtship,Propagation,Stable1);
 ASSIGN
        init(state) := Stable;
        next(state) := case
        state = Stable : (Fear, Goodwill, Stable);
        state = Fear : (Evasion, Changcolor, Hide);
        state = Goodwill : (Hide, Courtship, Stable1);
        state = Evasion : Stable1;
        state = Changcolor : Stable1;
        state = Hide : Stable1;
        state = Courtship : Propagation;
        state = Propagation : Stable1;
        1 : state;
 esac;

 DEFINE
        a := state = Fear;
        b := state = Goodwill;
        c := state = Courtship;
        d := state = Hide;
        e := state = Propagation;
```

Fig9: Verification result using SMV

4. Conclusion

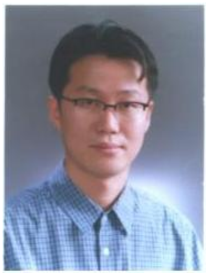
In this paper, (we defined the attribute value of objects for oceanic NPC models generation), and designed them with FSM algorithm. We also put the NPC models into the virtual ocean and realized that they act according to the attribute value in the virtual ocean. In addition, we could discover the error in the initial design by verifying FSM algorithm using SMV. This could reduce the burden of developers about the design becoming more complicated. Furthermore, we will need the formalizing and fractionalizing of the attribute for design and implementation of the more intelligent NPC models.

Acknowledgments

This research was supported by Ministry of Information and Communication (MCT), Korea, under the CRC (Culture Research Center) support program supervised by Korea Culture & Contents Agency (KOCCA) and DCRC (Digital Contents Cooperation Research Center).

References

[1] Incheol Kim, Jaeho Lee, “Architecture and Path-Finding Behavior of An Intelligent Agent Deploying within 3D virtual Environment”, Korea Information processing Society, Vol.10-1, No.1, 2003.2
[2] http://gamecode.org/article.php3?no=1588&page=0&current=ai&field=tip
[3] http://en.wikipedia.org/wiki/Non-player_character
[4] Timothy Kam, “Snthesis of Finite State Machines, Functional Optimization”, Kluwer Academic Publisher, Boston 1997.
[5] “Model Checking Large Software Specification”, IEEE Transactions on Software Engineering, Vol.24, No.7, July 1998.
[6] E. M. Clarke, O. Grumberg, and D. E. Long, “Checking and Abstraction”, In Proceedings of the Nineteenth Annual ACM Symposium on Principles of Programming Languages, January, 1992.
[7] Gihwon Kwon, TaeHo Eom, “Equivalence Checking of Finite State Machines with SMV”, Korea Information Science Society, Vol.30, No7, pp.642~648, 2003.8.
[8] David Gibson “Making simple work of complex functions”, SPLat Controls Pty Ltd, 1998-2000.
[9] K.McMillan, Symbolic Model Checking. Norwell, MA: Kluwer, 1993.
[10] T.S. Chow, “Test Design Modeled by Finite-State Machines,” IEEE Trans. Software Eng., vol. 4, no. 3, pp. 178-187, 1978
[11] S. Goren and F.J. Ferguson, “CHESMIN: A Heuristic for state Reduction in Incompletely Specified Finite State Machines,” Proc. 2002 Design, Automation, and Test in Europe Conf. and Exhibition, 2002



Chong Han Kim received the B.A degree in Mechanical Engineering from Chonnam National University in 1999 and the M.S. degree in Software Engineering in Chonnam National University in 2004. He has been working in Digital Contents Cooperation Research Center at Donshin Univirsity. His research fields are Formal Method and Digital Contents about Marine.

Seung-Moon Jeong received his B.S, M.S at Dongshin Univ, and Ph.D degrees in computer science from University of Donshin, NaJu, Korea, in 1991, 1999 and 2004. He taught digital contents at Dongshin University and researched Digital Contents Lab. He has been taught and researched as an instructor at Dongshin University. His research interests include Digital Contents, 3D animation, Image Processing, DRM, Ubiquitous Computing and VR.

Gi-Taek Hur received his B.S, M.S at Chonnam Univ, and Ph.D degrees in computer science from University of Kwangwoon, Seoul, Korea, in 1984, 1986 and 1994. He taught digital contents at Dongshin University and researched Digital Contents Lab. He has been taught and researched as a full professor at Dongshin University. His research interests include Digital Contents, 3D animation, Network Protocol, Image Processing, Fluid Animation, Ubiquitous Computing and RFID

Byung-Ki Kim received the B.S. and M.S. degrees in Department of Mathematics from Chonnam National University, Korea in 1978 and 1980. He received Ph.D. degree in Mathematics from Chonbuk National University, Korea in 2000. Since 1981, he is a professor of Computer Science at Chonnam National University. He is currently a senior vice-president in Korea Information Processing Society. His research interests include software engineering, formal method, object oriented methodology, and component based methodology.