가상환경 생성을 위한 인공생명 기반 진화엔진

Evolution Engine for Virtual Environment Generation based on Artificial Life

홍진혁, 조성배
연세대학교 컴퓨터과학과

Jin-Hyuk Hong, Sung-Bae Cho
Dept. of Computer Science,Yonsei University
E-mail : hjinh@candy.yonsei.ac.kr, sbcho@csai.yonsei.ac.kr

요 약

최근 컴퓨터 게임의 확산과 함께 보다 나은 가상환경 생성을 위한 기술에 대한 필요성이 증가하고 있다. 다양한 환경에서 지능적으로 행동하는 인공 캐릭터의 설계를 위해 다양한 인공지능 기술이 적용되고 있다. 하지만 게임의 캐릭터 설계에 적용된 휴리스틱이나 규칙기반 시스템 등의 기존 인공지능 기술은 게임 개발자에 의존적이기 때문에 플레이어가 쉽게 캐릭터의 행동패턴을 파악하여 게임의 흥미를 저하시키는 단점이 있다. 따라서 진화연산이나 신경망 등의 학습기반 인공지능 기술의 게임에의 적용이 모색되고 있다. 특히 진화를 이용한 지능기술은 자연계의 복잡성과 의외성을 모방하여 최적화된 지능보다는 속임수/의외성 등의 창의적인 지능행동의 생성을 가능하게하며 새로운 게임전략의 생성, 게임 캐릭터의 성격형성 및 다양한 행동 생성 등에 매우 유용하다. 본 논문에서는 진화기술의 게임에의 효과적인 적용을 위해 진화엔진을 설계 및 제작하고 인공지능 시뮬레이터에 적용하여 그 유용성을 확인하였다.

키워드: 게임, 인공지능, 창발성, 진화엔진

1. 서론

오늘날 컴퓨터의 보급과 동시에 무수히 많은 게임이 출시되고 있다[1]. 초기 하드웨어의 제한으로 그래픽 처리에만 한정되었던 게임기술은 최근 프로세서 자원이 풍부해지면서 다양한 인공지능 기술의 개발과 적용으로 옮겨지고 있다. 뿐만 아니라 플레이어는 보다 높은 수준의 지능을 가진 캐릭터를 선호하며, 인공지능은 그 게임의 성공여부를 결정짓는 중요한 요인이 되었다[1,2].

게임의 인공지능에 대한 관심이 높아지면서 기존의 그래픽 처리에 한정되었던 게임엔진의 개발이 점차 인공지능 처리에 초점 맞추어지고 있다. 설계 및 구현이 용이한 동시에 좋은 성능을 내는 유한상태기계, 사례기반추론, 퍼지논리 등의 인공지능 기술이 게임 캐릭터의 설계에 많이 사용된다[3]. 그러나 설계의 단순성으로 인해 플레이어에게 쉽게 캐릭터의 행동패턴을 파악한다는 단점이 있다[2,4]. 플레이어의 흥미를 유발하기 위해서 개발자는 다양한 종류의 행동규칙을 개발하여야 하는데, 이는 고비용, 저효율의 작업이다. 최근에는 신경망, 진화연산, 인공생명 등이 캐릭터 행동설계에 적용되어 기존 인공지능 기술의 한계를 극복하는 데 사용되고 있다[2,4,5]. 특히 진화연산은 다양하고 새로운 지능행동을 생성하여 여러 종류의 행동규칙을 가지는 게임 캐릭터를 제작할 수 있다는 장점이 있다.

본 논문에서는 게임 캐릭터의 행동 진화를 위해 진화엔진을 설계하고 구현하였다. 진화엔진은 범용의 진화구조를 제공하여 보다 쉽게 지능적인 캐릭터를 개발하고 새로운 게임에 적용되도록 도움을 준다. 게임 시뮬레이터에 적용하여 그 유용성을 확인하였다.

2. 관련연구

보통 게임에서의 인공지능은 캐릭터의 행동양식을 설계하는 데 사용되었다[1,3,4]. 전통적으로

유한상태기계, 사례기반추론, 결정트리, 신경망, 퍼지논리 등의 인공지능 기법이 이용되어 왔으나, 이들 대부분의 기법은 개발자가 행동규칙을 일일이 설계해야하는 어려움이 있다. 사람이 직접 모든 행동을 설계하면 다양성이 떨어지고 창조적인 행동이 나타나기 어렵다. 이러한 어려움을 극복하기 위해 보드 게임 등의 전통적 게임에서는 유전자 알고리즘 등의 기법이 적용되고 있지만, 기타 다양한 컴퓨터 게임에의 적용은 현실적으로 미비한 실정이다[5].

최근 캐릭터의 행동 설계를 위한 주요한 기법으로 인공생명 기술이 시도되고 있다[6,7]. 인공생명(Artificial life: A-life)은 생물체의 행동적 특성을 모방한 시스템에 관한 연구로, 하위 수준의 동작구조 간의 상호작용의 결과로 상위 수준의 기능이 생성되는 '창발적 행동'을 중요시한다. 인공생명 기술은 캐릭터의 행동을 결정하는 거대한 AI를 작은 단위로 분할하여 복잡하고 전역적인 행동을 하위 수준의 단순한 규칙들의 상호작용으로 생성한다. 이 때 효율적이고 우수한 상호작용이 이루어지도록 유전자 알고리즘을 사용하여 하위 규칙의 최적조합을 탐색하기도 한다.

3. 진화엔진

진화엔진은 비슷한 캐릭터 설계 시 재사용성을 제공하고 특정 게임에 국한된 것이 아닌 범용적인 사용이 그 장점이다. 다양한 상황에 적용되기 위해서는 진화엔진의 선택사항이 필요하다. 본 논문에서 개발한 진화엔진은 그림 1에서처럼 대표적인 진화연산인 유전자 알고리즘[8]과 유전자 프로그래밍[9]을 주요 동작 모듈로 하였다. 유전자 알고리즘은 고정적인 표현형의 염색체를 이용하여 캐릭터의 행동을 설계하며, 유전자 프로그래밍은 가변적인 표현형의 염색체를 이용하여 캐릭터 행동을 설계한다.

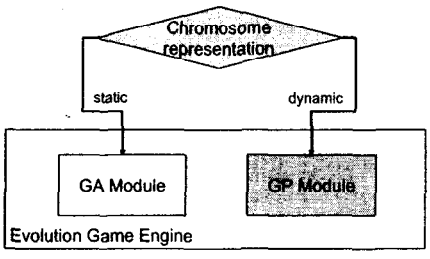

그림 1. 진화엔진의 구성

3.1 GA 모듈

GA모듈은 그림 2와 같이 하나의 개체를 정의할 수 있는 Individual controller(chromosome)와 전체 진화동작을 설계하는 GA controller로 구성되어 있다. Individual controller에서는 캐릭터의 행동양식을 염색체로 정의하며 한 개체의 적합도를 평가하는 적합도 함수를 제작한다. GA controller에서는 진화 과정에 대한 실험 설계를 한다. 즉, 최대 진화 세대수, 집단의 크기, 교차율, 돌연변이율, 선택방법, 교차방법 등에 대해서 결정을 한다. 사용자가 GA모듈을 사용하기 위해서는 먼저 individual controller에서 개체의 적합도를 평가하는 적합도 함수를 직접 제작하여야 하고, GA controller이 수행되도록 실험의 파라메터를 설정하고 run()함수를 실행하면 된다.

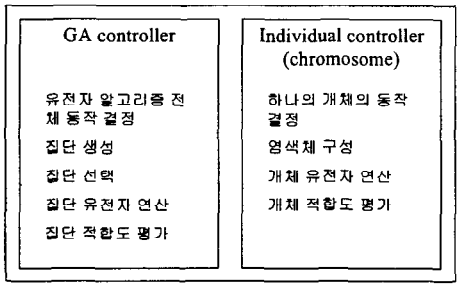

그림 2. GA모듈 구성

사용자는 다음의 단계에 따라 진화엔진의 GA모듈을 이용할 수 있다.

- 염색체 설계
- 적합도 함수 제작
- 각종 진화 파라메터 설정

캐릭터의 행동을 보다 유연하게 정의하기 위해 GA모듈의 염색체는 정수형 배열로 되어 있다. chromosome.string 변수를 통해 설계되는 캐릭터의 행동진화를 위한 염색체는 다음과 같이 입력부와 출력부로 구성된다.

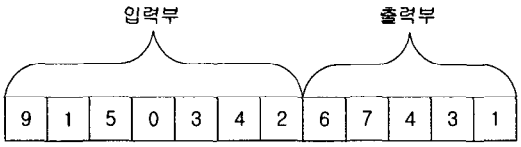

그림 3. 염색체 구성

입력부는 캐릭터가 게임으로부터 획득하는 정보를 표현하고 출력부는 캐릭터가 취할 행동을 나타낸다. 염색체 설계가 끝나면, 적합도 함수를 설계하는데, 보통 게임을 수행한 결과가 적합도가 된다. 이 두 단계는 Individual controller에서 사용자가 직접 설계하여야 하는 부분이며, 이후에는 GA controller에서 진화를 위한 파라메터를 설정함으로 진화실험 준비가 완료된다. 다음은 GA모듈의 주요한 함수들을 설명한 것이다.

- popGeneration() : 집단의 메모리 할당
- popInitialization() : 집단 초기화
- popFitnessEvaluation(): 집단 적합도 평가
- run() : 진화 세대 수행 함수
- popChange() : 세대 교체 수행 함수

- setPopSize() : 집단 크기 설정
- setPopGeneLength() : 염색체 길이 설정
- setGenerationLimit() : 최대 세대 수 설정
- setSelectionRate() : 선택율 설정
- setCrossoverRate() : 교차율 설정
- setMutationRate() : 돌연변이율 설정
- setFitnessFunction(float (*fn)(int*, int)): 사용자 설계 적합도 함수

보통 캐릭터의 적절한 행동양식을 탐색하기 위한 공간은 매우 광범위하며 해영역이 매우 불규칙하다. 보통 적합도 함수를 설계하기가 매우 어려고, 잡음도 많다. 이에 대해 유전자 알고리즘은 해 탐색 영역이 매우 큰 경우에 다른 알고리즘에 비해 매우 우수한 성능을 가지며 잡음에도 강한 특성이 있다. 이 외에, 유전자 알고리즘은 다양한 최적해의 검색 가능성이 있기 때문에 다양하고 창발적인 캐릭터 행동양식의 생성을 가능하게 한다[7,10,11].

3.2 GP 모듈

GP모듈은 GA모듈과 동일한 형태로 하나의 개체를 정의할 수 있는 Individual controller (chromosome)과 전체 진화동작을 설계하는 GP controller로 구성되며, 클래스는 ControllerGP class와 NodeGP class가 있다. GP모듈은 GA모듈과 염색체의 표현형이 다르게 트리구조를 이용한다. 염색체는 그림 4와 같이 산술연산자(+, -, *, /, sin, cos, %)와 입력변수 및 상수로 구성되는 트리구조를 가진다. 따라서 입력변수 사이의 산술적 관계를 도출할 수 있다. 적합도 함수는 GA모듈과 비슷한 형식으로 게임의 결과를 적합도로 사용하지만, 트리의 계산값을 바탕으로 캐릭터를 움직이고 이에 대한 결과를 적합도로 구하게 된다.

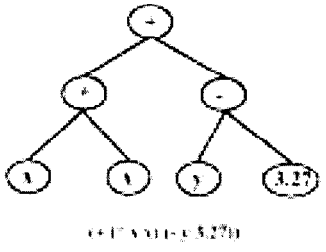

그림 4. 유전자 프로그래밍의 표현형인 트리구조

유전자 프로그래밍이 유전자 알고리즘과 다른 점은 유전자 알고리즘이 표현형으로 고정된 길이의 bit 배열을 사용하는 것에 반해 유전자 프로그래밍은 트리구조를 사용한다. 동적인 유전자 표현형으로 보다 폭 넓은 해 탐색 영역을 가지며, parse-tree 형태로 개체를 표현하기 때문에 명시적인 규칙의 진화에 매우 유리하다[9].

4. 게임에의 적용

본 논문에서는 최근 관심을 모으고 있는 인공지능 전투 시뮬레이터인 로보코드를 바탕으로 유전자 알고리즘을 이용한 캐릭터 행동의 진화를 진화엔진으로 구현함으로써 진화엔진의 캐릭터 행동진화가 얼마나 유용한지 검증하였다.

그림 5. 로보코드

로보코드는 IBM Alphaworks에서 개발한 자바기반 탱크 전투 시뮬레이터로서, 사용자에 의해 프로그래밍되는 탱크가 상대방 탱크의 공격과 벽을 피해 다니며 공격을 한다[12]. 일반적으로 탱크는 사용자의 설계와 프로그래밍으로 제작되어지기 때문에 승리를 위해 정교한 행동양식이 설계되지만 행동패턴이 단순하기 쉽다.

본 논문에서는 탱크의 행동을 크게 이동전략, 발사전략, 총알세기전략, 레이더탐색전략, 타겟선택전략으로 구분하고 아래 표와 같이 각각 3~8개의 세부행동양식을 정의하였다.

이동	발사	총알세기	레이더탐색	타겟선택
임의이동	현재위치 예측	거리에 따라	항상돌리기	약한로봇 우선
단순선형 왕복	선형예측	약하고빠름	타겟집중	집중사격
임의선형 왕복	복합선형 예측	강하고느림	타겟범위 집중	방어형집중
단순원형 왕복		중간세기로		가까운로봇
임의원형 왕복		명중률에 따라		
반중력운동				
정지				
총알회피				

진화엔진을 이용하여 기본행동의 최적의 행동조합을 탐색하고 얻어진 조합을 캐릭터의 행동양식으로 설정하였다. 염색체 설계는 특별한 입력부가 필요없기 때문에 그림 6과 같이 출력부만 갖는 25(5*5)자리 정수형 배열을 사용하였다. 각 자리는 상위행동전략을 가리키고 그 값은 세부전략을 나타낸다. 미리 정의된 5가지 이벤트에 따라 해당하는 전략조합을 수행한다.

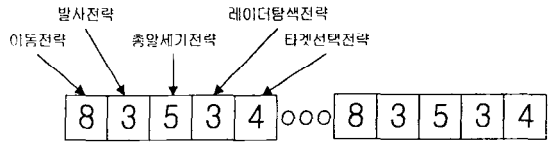

그림 6. 로보코드를 위한 염색체 설계

적합도 함수는 로보코드 전투 결과로, 탱크가 획득한 점수를 사용하였고, 대상로봇으로는 로보코드에서 샘플로 제공하는 CornerRobot, FireRobot, WallRobot을 선택하였고, 그 외 2002년도 Robocode Rumble에서 수상한 탱크인 BigBear를 사용하였다. 집단 내 개체 수는 25, 최대 진화 세대는 50으로, 기타 각종 파라메터는 다양하게 설정하여 진화를 수행하였다.

아래 표는 각 탱크에 대한 진화 결과로 다양한 종류의 우수한 행동양식이 생성되었음을 확인할 수 있으며, 또한 생성된 행동양식은 각각 대상 로봇에 대해 다르다는 것을 확인할 수 있다. 이것은 진화엔진이 각 대상에 적합한 캐릭터 행동양식을 생성한다는 것을 보여준다.

대상 탱크	우수한 행동양식 (150 세대)	최대 빈도수 행동 부분	
		기본	이벤트 발생시
Corner	23	000310	001003
Fire	13	000200	012413
Wall	7	400302	412223
BigBear	0 (4)	000313	011112

아래 표는 WallRobot에 대해 생성된 창발적 행동전략을 정리한 것이다. 동일한 상대 로봇에 대하여서도 염색체 정보가 다르면서 우수한 전략들이 생성되었다. 또한 몇몇 행동양식들은 매우 흥미로운 것으로, 사전에 예상하지 못한 것도 있었다.

창발적 행동	내용	염색체
람보	무차별 난사	012312
사격수	적 탱크의 추적 및 정교한 공격	412422
박치기 왕	적 탱크에 박치기 시도	401022
도망자	적 탱크의 공격을 피하다가 적 탱크 에너지가 떨어지면 공격	210003
종횡무진 활주	전투장을 정신없이 돌아다님	012112

5. 결론 및 향후과제

진화연산은 기존의 개발자 의존적인 인공지능 기술이 가지는 개발상의 한계를 극복하여 다양하고 흥미로운 캐릭터를 생성하도록 한다. 진화를 통해 얻어진 캐릭터 행동의 창발적 요인은 게임의 속임수/의외성 등의 특성을 만족시키고, 특정 상대에 대해 다양한 전략을 생성하여 게임을 진행하면서 새로운 흥미를 사용자에게 제공할 수 있다. 이를 위해 본 논문에서는 진화기술이 게임에 효과적으로 적용될 수 있도록 진화엔진을 제작하고 인공지능 시뮬레이터인 로보코드에 적용하여 그 유용성을 확인하였다. 향후에는 다양한 게임 플랫폼에 진화엔진을 적용하여 게임에의 진화기술 적용가능성을 분석하고자 한다.

감사의 글

본 논문은 한국문화콘텐츠진흥원의 지원에 의한 것임.

참고문헌

[1] S. Woodcock, "Game AI: The state of the industry," *Game Developer Magazine*, August 1999.

[2] J. Laird and M. Lent, "Human-level AI's killer application: Interactive computer games," *AI Magazine*, vol. 22, no. 2, pp.15-26, 2001.

[3] D. Johnson and J. Wiles, "Computer games with intelligence," *IEEE Int. Fuzzy Systems Conf.*, pp. 1355-1358, 2001.

[4] T. Revello and R. Mccartney, "Generating war game strategies using a genetic algorithm," *Proc. of the 2002 Congress on Evolutionary Computation* , pp. 1086-1091, 2002.

[5] K. Chellapilla and D. Fogel, "Evolution, neural networks, games, and intelligence," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1471-1496, 1999.

[6] C. Langton, C. Taylor, J. Farmer, and S. Rasmussen, "Artificial life 2," *in Sante Fe Institute Studies in the Sciences of Complexity,* Addison-Wesley, pp. 511-547, 1992.

[7] M. Mitchell and S. Forrest, "Genetic algorithms and artificial life," Artificial Life, vol. 1, no. 3, pp. 267-289, 1994.

[8] D. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley Publising Co. Inc., 1989.

[9] J. Koza, "Genetic programming," Encyclopedia of Computer Science and Technology, vol. 39, pp. 29-43, 1998.

[10] K. Sims, "Evolving virtual creature," Proc. of the 21st Annual Conference on Computer Graphics and Interactive Techniques, pp. 15-22, ACM Press, 1994.

[11] C. Reynolds, "Competition, coevolution and the game of tag," *Proc. of the 4th Int. Workshop on the Synthesis and Simulation of Living Systems*, pp. 59-69, 1994.

[12] S. Li, Rock 'em, sock 'em robocoe!, 2002.http://www-106.ibm.com/developerworks /java/library/j-robocode