물고기 무리짓기 구현

유현지*, 박종호*, 이면재*
*백석대학교 정보통신학부
e-mail:87fbguswl@hanmail.net*, pjh8501@naver.com, davidlee@bu.ac.kr

Implementation of Fish Schools

Hyeon-Ji Ryu*, Jong-Ho Park*, Myoun-Jae Lee*
*Division of Information Communication, Baekseok University

요 약

무리짓기 시뮬레이션은 무리를 짓는 NPC의 인공 지능을 구현하는 것으로 게이머들에게 실세계와 같은 몰입에 도움을 준다. 이를 위하여 본 논문은 분석된 실세계에서의 물고기 무리짓기의 행동 패턴을 오우거 엔진을 이용하여 구현[1]하고, 이를 실세계의 물고기 떼의 행동 패턴과 비교하였다.

1. 서론

게임에서 몬스터 NPC들의 일정 던전안에서 움직임을 통해서 모여있다가 흩어짐을 보여주거나, 몬스터 NPC와의 전투중 싸움을 하지 않는 근처의 몬스터가 다가와 난입을 통해서 무리짓기를 보여준다. 대규모 전투를 할 때 용병들을 전형에 맞춰서 정렬을 시키기도 하고 사용자 캐릭터가 다리를 건너기, 다른 곳으로 이동을 할 때 이용한다[4][5].

[그 림1] 게임에서 무리짓기

그러나 이러한 무리짓기는 NPC들이 동일하게 작동하여서 게임의 재미를 감소시킬 수 있다.

본 논문은 실세계의 무리짓기를 게임에 적용하기 위하여 무리짓기 알고리즘[2][3]을 이용하여 무리짓기 시뮬레이션을 하고, 이를 실세계 연구와() 비교한다. 이 시뮬레이션을 위하여 Ogre 게임 엔진을 이용하였고, 물고기의 무리짓기에 대한 규칙은 물고기의 회피영역에 다른 물고기가 들어와 충돌 할 수 있는 경우 최소한의 거리를 유지하기 위한 충돌 회피, 물고기 응집영역에 다른 물고기가 들어올 경우 무리를 지어 움직이기 위한 응집, 정렬 규칙을 사용한다.

본 논문의 구성은 다음과 같다. 2장에서는 시뮬레이션을 위해 사용된 게임엔진과 무리짓기 규칙에 대해 설명한다. 3장에서는 무리짓기에 의한 결과를 기술하고 이를 실세계의 무리짓기 결과와 비교한다. 4장에서는 실세계와 시뮬레이션의 비교를 분석하고 게임 적용에 대해 기술한다.

2. 본론

2.1 Ogre 엔진 소개

Ogre 엔진은 Object oriented Graphic Rendering의 약자로 그래픽 랜더링과 에니메이션을 수행하는 공개 엔진으로 물리엔진과 AI엔진을 포함하는 Full 3D 엔진이 아니다. 이는 C++기반의 객체지향 엔진으로 DirectX와 openGL을 동시 지원하고 다양한 플랫폼을 가진다.

Ogre 엔진은 공개 엔진이지만 상용개발이 가능하고 라이센싱 모델은 GNULGPL이다.

Ogre엔진 구조는 하나의 Root를 중심으로 장면관

리, 자원 관리, 랜더링 3부분으로 나뉜다. 장면관리는 전체적으로 모니터에 출력되는 장면을 관리한다. 장면 관리자는 노드관리, 객체관리, 재질관리가 있다. 자원 관리는 객체에 입힐 메쉬, 이미지, 모델을 전반적으로 관리 한다. 랜더링은 자원관리와 장면관리에 의해 설정된 객체 내용을 화면에 출력하는 일을 한다.

2.2 무리짓기 규칙

무리짓기는 1987년 Craing Raynolds가 발표한 논문에 소개된 기법으로 물고기 떼와 같은 무리의 집단 행동을 보이도록 하는 규칙이다. Raynolds는 무리짓기 행동을 보이는 생물을 보이드(boid)라 명명하고 무리짓기 규칙을 조타행동(steering behavior)라 명명했다[3][4].

조타행동에는 충돌회피와 정렬이 있다. 각 보이드는 다른 보이드와의 조타행동을 결정하기위해 2개의 거리영역을 설정하였다. 보이드와 보이드의 충돌을 막기위한 회피영역과 보이드간 응집 및 정렬을 위한 상호작용이 있는데 이 영역은 충돌회피 영역보다 넓다.

충돌 회피 영역은 δ, 상호작용 영역은 P, 보이드의 방향은 $V_i(t)$, 보이드의 위치는 $C_i(t)$, 외부자극에 대한 보이드의 응답 대기 시간은 $\triangle t$, 보이드의 이동 방향을 $d(t)$로 표시한다.

*충돌 회피

보이드가 다른 보이드와의 충돌을 피하고 일정 거리를 유지하도록 하는 규칙이다. 보이드의 회피영역 δ에 다른 보이드가 위지해 있을 때, 보이드는 이 영역에 있는 보이드와의 관계를 계산해 방향을 전환하여 충돌을 피하고 보이드간의 일정한 거리를 유지한다. 보이드가 다른 보이드간의 거리 유지를 위한 새로운 이동 방향을 구하는 식은 식(1)과 같다.

$$d_i(t+\triangle t) = -\sum_{j \neq i} \frac{c_j(t) - c_i(t)}{|C_j(t) - C_i(t)|} \quad ...(1)$$

*정렬

보이드의 상호작용 엉역 P안에 다른 보이드가 위치해 있을 때, 그 보이드와 응집하여 무리를 짓고 같은 방향과 속드를 갖도록 하는 규칙이다. 이 규칙은 회피 영역 δ에 어떠한 보이드도 존재하지 않고 영역 P에만 보이드가 있을 때 보이드가 반응을 한다. 보이드간 정렬을 하기위한 방향을 구하는 식은 식(2)와 같다.

$$d_i(t+\triangle t) = 1/2[\sum_{j \neq i} \frac{c_j(t) - c_i(t)}{|c_j(t) - c_{i(t)}|} + \sum_{j=i} \frac{v_j(t)}{v_i(t)}] \quad ..(2)$$

보이드의 회피영역 δ과 상호작용 영역 P에 다른 보이드가 없을 때, 보이드는 식(3)과 같이 새로운 방향 벡터를 자신이 마지막에 갖고 있던 방향 벡터값을 갖는다.

$$d_i(t+\triangle t) = V_i(t) \quad ...(3)$$

보이드의 새로운 위치는 위의 3가지 규칙에 의해 구해지 방향 벡터를 이용하여 구한다. 식(4)는 보이드의 새 위치를 구하는 식이다.

$$c_i(t+\triangle t) = c_i(t) + V_i(t+\triangle t)\triangle t * s \quad ..(4)$$

이 때, $C_i(t+\triangle t)$는 보이드의 새 위치값, $V_i(t+\triangle t)$는 새 방향벡터, S는 속도,$\triangle t$는 반응시간을 나타낸다.

3. 구현 및 실제 무리짓기와 비교

3.1 무리짓기 시뮬레이션 환경

실험은 보이드의 수와 어항의 크기에 변화를 주어 시뮬레이션 하였다. 보이드의 수는 어항의 크기를 x축 100~-100, Z축 100~-100의 조건에서 각각 10마리, 20마리 일 때를 시뮬레이션 했다. 어항의 크기는 보이드 10마리의 고정 조건에서 x축이 100~-100, z축 100~-100 일 때와 x축 200~-200, z축 200~-200일 때를 시뮬레이션 했다.

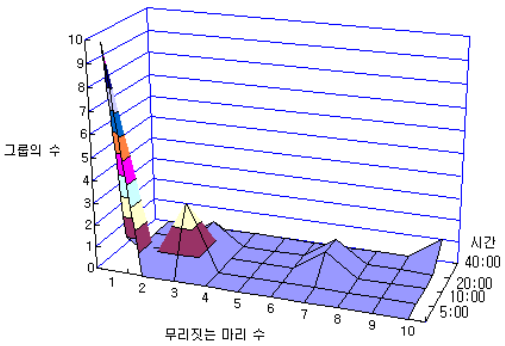
나머지 조건들은 항상 같은 값으로 설정해 놓았다. 응답 대기시간 $\triangle t$는 0.1, 보이드의 속도 s는 1.25, 보이드의 크기 4cm, 회피영역 δ는 4BL, 응집영역 P는 8BL로 설정하였다. 이때 1BL은 4cm이다. 초기의 물고기의 위치와 방향은 랜덤하게 주어진다.

무리짓기 시뮬레이션은 Ogre 엔진 1.4.6버젼을 사용하고, 컴파일러는 Visual C++ express 2005를 사용하였다.

3.2 무리짓기 결과

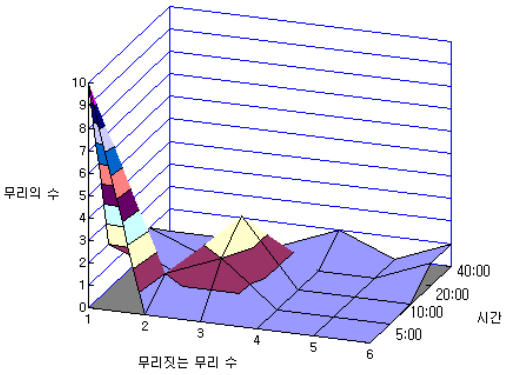
[그림 2]는 보이드가 10마리, 어항의 크기가 X축 100~-100, Z축 100~-100일 때의 시간경과에 따른 보이드 그룹의 변화를 나타낸다. [그림2]를 보면 초반에는 각자 움직이다가 2~3마리씩 응집하여 무리지어 움직이고 시간이 경과할수록 무리짓는 보이드의 수가 늘어남을 볼 수 있다. 40분이 경과 했을 때는 10마리의 보이드가 하나의 무리를 지어 이동하는 것을

보여준다.



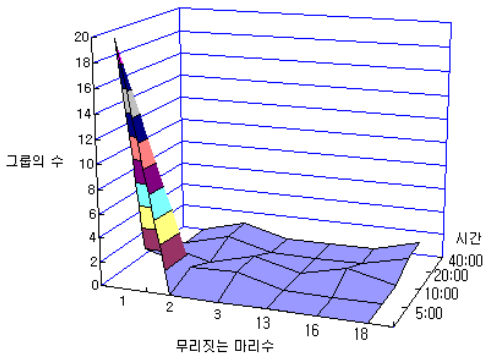
[그림 2] 보이드 10, 어항크기 x,z축 100~-100일 때

[그림 3]은 보이드가 10마리, 어항의 크기가 X축 200~-200, Z축 200~-200일 때 보이드 그룹의 변화를 보여준다. 초반의 보이드 2~3마리씩 무리지어 움직이는 것을 보여준다. 그러나 [그림 2]처럼 시간 경과에 따른 무리짓는
그룹의 수가 줄어들고, 한 그룹 내의 보이드 수가 늘어나지 않는다. 이는 어항의 크기가 [그림2]보다 4배 커지면서 보이드가 초반 2~3마리씩 무리지은 그룹이 시간이 경과해도 비슷하게 움직인다. 또한 [그림3]을 통해서 보면 [그림2]와 다르게 40분 경과해도 보이드가 하나의 그룹으로 움직이지 않고 최대 6마리가 하나의 그룹을 이루는 것을 볼 수 있다.



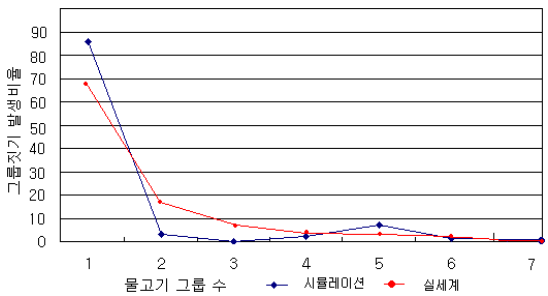
[그림3]보이드 10, 어항의 크기 x,z 200~-200일 때

[그림 4]는 보이드가 20마리, 어항의 크기가 X축은 100~-100, Z축 100~-100일 때 보이드의 변화를 보여준다. [그림4]는 보이드의 수가 많고, 어항의 크기는 작기 때문에 시작하자마자 보이드들 간의 응집을 통해서 2마리씩 응집해 다니기 시작하고, [그림2]에서보다 빠르게 보이드들이 응집하여 무리를 지어 움직이는 것을 볼 수 있다.



[그림4] 보이드 20, x,z축 100일 때

[그림5]는 실세계의 무리짓기 실험을 한 결과와 Ogre 엔진을 이용하여 무리짓기 시뮬레이션 한 결과를 비교한 그래프이다. [그림5]는 D.J. HOARE et al.[3]이 물고기 무리짓기 관찰 조건과 동일하게 수행했다. [그림5]는 [그림2]의 보이드의 조건과 동일하고 관찰은 30초 간격으로 60분간 관찰하였다. [그림5]의 그래프는 시간 관계를 보는 것이 아닌, 60분간 물고기 무리짓는 수와 그 무리의 그 비율을 나타낸 그래프이다.



[그림 5]실세계와 시뮬레이션 비교

3.3 시뮬레이션과 실세계의 패턴 비교

시뮬레이션을 통해서 초반의 물고기들이 각각 움직이다가 두세 마리씩 무리를 지어 움직이게 되고, 시간이 흐를수록 하나의 무리를 이루는 것을 볼 수 있다. 보이드의 무리짓기가 한 마리 혼자 다닐 때와, 2마리씩 무리를 이루어 다니는 비율의 차이가 있지만, 한 그룹의 보이드 수가 많아질수록 그룹의 무리짓는 비율이 적어지는 실세계의 결과와 시뮬레이션의 결과가 비슷함을 볼 수 있다.

4. 결론 및 추후 연구방향

본 논문은 Ogre 게임 엔진을 이용하여 응집 회피의 보이드의 무리짓기를 이용한 시뮬레이션을 실세

계 보이드의 무리짓기와 비교하였다.

실세계의 보이드 무리짓기에서 그룹짓는 수의 비율과 시뮬레이션으로 얻은 비율이 비슷한 결과를 얻었다. 이러한 결과는 보이드의 무리짓기를 이용하여 게임을 제작하면 보다 현실에 가까운 게임 환경을 사용자에게 제공하여 몰입을 증진 시킬 수 있을 것으로 보인다.

5. 참고 문헌

[1] 유현지, 이면재, “Ogre 엔진을 이용한 물고기 떼 시뮬레이션”, 한국산학기술학회, 제10권, 제2호, pp782-784, 12월, 2009

[2] PETER H.WERGE, MARTIN WIKESKI, et.al, "ANTIBIRDS PARASITIZE, FORAGING ARMY ANTS", Ecology, 86(3). 2005, pp.555-559.

[3] D.J.HOARE, I.D. COUZIN, J.-G.J. GODIN &J.KRAUSE "Context-dependent group size choice in fish", Elsevier Ltd. ANIMAL BEHAVIOUR,67, pp.155-164,2004

[4] Steven Woodcock, 플로킹: 집단 행동을 흉내내는 간단한 기법, Game Programming Gems 1, pp.401-415, 2001.

[5] Steven Woodcock, 먹고 먹히는 플로킹: 포식자와 먹이, Game Programming Gems 2, 이 pp.423-430, 2002.