Ogre 엔진을 이용한 물고기 떼 시뮬레이션

유현지* , 이면재*
백석대학교 정보통신학부
87fbguswl@hanmail.net*, davidlee@bu.ac.kr

Simulation of Fish School using Ogre Engine

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

요 약

무리짓기를 통한 인공지능 구현은 게이머들에게 실세계와 같은 느낌을 제공한다. 본 논문에서는 오우거 엔진을 이용하여 실세계의 물고기 떼의 무리 짓기 규칙을 응집, 회피, 정렬을 중심으로 시뮬레이션 한다. 구현 결과, 본 시뮬레이션은 실세계에서의 무리짓기에서와 같은 회피, 정렬의 특징을 보여준다.

1. 서론

현재 온라인 게임에서는 실세계에서와 같은 환경을 구성하여 게이머에게 실제감을 제공하려는 노력이 많이 시도되고 있다. 이중 무리짓기는 대규모의 NPC들이 이동하거나, 게임 캐릭터를 공격하거나, 도망가는 것에 많이 사용된다. [그림 1]은 게임에서의 무리짓기를 보여준다.

[그림 1] 게임에서의 무리짓기

실세계에서의 무리짓기는 새들이 대열을 이루어 목적지를 향해 비행하거나, 물고기들이 일정한 패턴으로 이동하거나, 개미들이 무리를 지어 이동하는 등의 형태들이 많이 있다. 이러한 실세계에서의 무리짓기에 관한 연구에는 개미[1], 물고기[2] 등이 있으며 생물학적 관점으로 많이 진행되어지고 있다. [그림 2]는 실세계에서의 물고기떼의 무리짓기를 보여준다.

[그림 2] 실세계 물고기의 무리짓기

본 논문은 실세계의 무리짓기를 게임에 적용하기 위한 시도이다. 이를 위하여 오우거 엔진을 이용하여 물고기떼의 무리짓기[2]의 연구를 시뮬레이션 한다. 이때 사용되는 무리짓기 규칙들은 물고기들의 개체 사이에 최소 거리를 유지하는 충돌 회피, 지역적인 상호 작용 영역안에 있는 모든 다른 물체들에게 반응하는 정렬 법칙으로 구성된다.

본 논문의 구성은 다음과 같다. 2장에서는 오우거 엔진의 특징과 무리짓기에 관한 관련 연구를 기술하고, 3장에서는 실험 결과를 기술한다. 4장에서는 결론 및 추후 연구 방향을 기술한다.

2. 본론

2.1 Ogre 엔진 소개

Ogre는 Object oriented Graphic Rendering Engine의 약자로 그래픽 랜더링 및 애니메이션을 수행하는 C++기반의 객체지향 게임 엔진이다. Ogre는 2001년 영국의 Steve'sinbad' streeting이 개발한 공개 엔진

으로 Direct3D와 OpenGL을 동시지원을 하고, 다양한 플랫폼을 갖는다.

Ogre 엔진의 구조는 하나의 Root를 중심으로 장면 관리, 자원관리, 랜더링, 3개의 부분으로 이루어져 있다. 장면 관리자는 전체적으로 화면에 출력되는 장면들을 관리한다. 이 장면 관리자는 노드관리, Object 관리, 재질관리가 있다. 자원 관리자는 Object에 입혀질 Mesh나 이미지, 모델에 대한 데이터를 종합적으로 관리한다. 마지막으로 랜더링은 자원 관리와 장면 관리에 의해 설정된 Object의 내용을 화면에 출력해 주는 일을 한다.

Ogre 엔진은 다양한 플랫폼에서 지원이 가능하고, 객체 지향 인터페이스 방식으로 간결한 엔진 클래스와 인터페이스를 제공하고, 소스 구현이 간단하고 쉽다.

Ogre 엔진은 다른 게임 엔진과 다르게 애니메이션, 그래픽 랜더링만 지원하는 엔진으로 물리엔진이나 AI 엔진을 포함하는 Full 3D 엔진이 아니다. 공개 엔진인 Ogre는 상용개발이 가능하다. Ogre 엔진의 라이센싱 모델은 GNNLGPL이다.

2.2 무리짓기 규칙

무리짓기는 플로킹(flocking)이라 하여 1987년 Craing Raynolds 가 발표한 논문에서 처음 소개된 기법으로 물고기 떼와 같은 무리가 집단행동을 보이도록 만든 3가지 규칙이다. Raynolds는 이 규칙과 관련되어, 비슷한 집단 행동을 보이는 파리떼, 새 떼, 물고기 떼와 같은 생물을 보이드(boid)라 명하고, 무리짓기 규칙을 조타행동(steering behavior)이라 명명하였다. 3가지의 무리짓기 규칙은 충돌 회피, 정렬 규칙이다.

● 충돌회피

충돌 회피는 보이드가 다른 보이드와의 충돌을 피하고 일정한 거리를 유지하도록 하는 규칙으로 보이드의 회피영역에 다른 보이드가 들어오게 되면 충돌을 피하기 위해서 보이드가 회피영역 안에 들어온 다른 보이드와의 방향을 계산하여 해당 보이드의 방향을 변경시켜서 다른 보이드와의 일정거리를 유지한다. 식(1)은 다른 보이드와의 최소 거리 δ에 다른 보이드가 있을 경우에 보이드의 방향을 보여준다. 각 보이드의 위치는 $C_i(t)$, 외부 자극에 보이드가 무리짓기 응답 대기시간 간격을 △t, 물고기가 이동하려는 방향을 d(t)로 나타낸다.

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

● 정렬

보이드가 주변의 다른 보이드와 동일한 방향과 속도를 유지하도록 하는 규칙으로 정렬은 보이드가 다른 보이드와 관계를 맺는 상호작용영역에 보이드들이 들어와 있을 때, 해당 보이드들과 무리를 지어 하나의 보이드가 움직이듯이 보여지는 규칙이다. δ안에 다른 개체들이 없는 경우에 상호 작용(응집) 영역 ρ에 있는 개체에 반응한다. 식(2)는 이를 나타낸다. 이때 $V_i(t)$는 보이드의 방향을 나타낸다.

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

상호작용영역 ρ에 다른 보이드가 없을 때, 보이드는 자신이 갖고 있는 방향벡터를 그대로 사용하여 이동한다. 식(3)은 이를 나타낸다.

$$d_i(t+\triangle t) = v - i(t) \quad ....(3)$$

보이드의 최종 위치는 3가지 규칙에 의해서 구해진 새로운 방향벡터를 가지고 새로운 위치를 얻게 된다. 식(4)을 이용하여 보이드의 새 이동 좌표를 구한다.

$$c_i(t+\triangle t) = c_i(t) + v_i(t+\triangle t)\triangle t * s \ldots \quad (4)$$

새로운 위치 벡터 $C_i(t+\triangle t)$는 보이드의 이전 위치벡터 $C_i(t)$에 다른 보이드와의 관계에 의해 새로 구해진 방향 벡터 $V_i(t+\triangle t)$와 보이드의 속도 s, 그리고 보이드의 반응시간 ($\triangle t$)을 곱한 값과 더해서 얻어진다.

3. 실험 결과

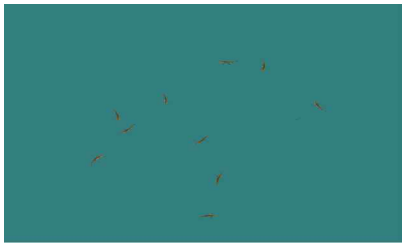
3.1 시뮬레이션 환경

Ogre 엔진 SDK 버전은 1.46을 사용하고 컴파일러는 Visual C++ express 2005 버전을 사용한다. 실험 컴퓨터는 CPU는 2.53GHz, RAM은 2GB이고 그래픽 메모리 사양 512MB이며 사용된 운영체제는 Microsoft XP Pro이다.

실험에 사용된 파라미터는 응답 대기시간△t는 0.01, 이동속도 s=1.25BL/s, 물고기의 크기는 4cm로 설정한다. 여기에서 4Cm는 1BL이 된다. δ는 4L, p=8L로 설정한다. 물고기 떼의 활동 범위는 x축으로 100~ -100 (200), Z축으로 100~-100(200)으로 정사각형의 수족관을 만들어 준다. 10마리의 물고기를 실험에 사용한다. 이 때에 물고기의 위치와 뱡향은 랜덤하게 준다.

3.2 시뮬레이션 결과

[그림 3]과 [그림 4]는 실험 초기 화면과 정렬된 물고기 떼를 보여준다. 정렬된 물고기 떼는 일정 시간이 지난 후의 화면이다. 시뮬레이션 초기 화면은 물고기의 방향이 서로 다르고 정렬되어 있지 않음을 보여준다. [그림 4]는 화면의 오른쪽 2마리씩 물고기들이 부분적으로 정렬한 모습을 보여준다.



[그림 3] 시뮬레이션 초기화면

[그림 4] 정렬 화면

두 번째 실험은 물고기의 크기를 원래의 mesh 보다(4cm) 2배 키우고 수족관의 범위를 가로 200, 세로 200으로 한 정사각형의 수족관을 설정하고, 모든 물고기들에 동일한 속도 1.3s, $\triangle t$는 0.01로 설정하고 물고기에게 방향과 위치만을 랜덤하게 부여한다.

실험 결과, [그림 3]과 [그림 4]와 동일하였는데, 다만 좀 더 빠르게 물고기들이 응집되었다 분리되는 것을 확인 할 수 있었고, 물고기가 커지면서 물고기의 분리규칙에 의한 다른 물고기를 피해가는 이동 경로가 더욱 자세하게 보인다.

실험 초기에는 물고기 각각의 방향에 의해서만 이동을 하다가 얼마 후 물고기들끼리 응집하고 무리지어 이동하거나, 너무 가까이 응집하여 분리되어 이동하는 것을 확인 할 수 있었다.

그러나 물고기가 목적지 없이 움직이기 때문에 응집이 되기가 쉽지 않았고, 정렬되어 이동하는 모습도 다양하지 않았다. 주변에 물고기가 모이지 않는 물고기는 무한정으로 양쪽의 벽만을 왔다 갔다 하는 모습을 보이면서 다른 물고기와의 응집이 되지 않았다.

4. 결론 및 추후 연구결과

본 논문은 Ogre 엔진을 이용하여 Raynolds가 제안한 무리짓기 3가지 규칙인 응집, 정렬 규칙을 중심으로 물고기 떼의 무리짓기를 시뮬레이션하였다.

보이드의 목적지가 없어서 정렬과, 응집이 어려웠는데, 보이드에 목적지만 설정해 준다면, 보다 실세계와 유사한 물고기의 무리짓기를 보여 줄 것으로 보인다. 추후에는 다양한 물고기 수와 환경, 그리고 실지 게임에서 사용되는 무리짓기 알고리즘과 성능을 비교 평가할 예정이다.

참고 논문

[1] PETER H.WERGE, MARTIN WIKESKI, et.al, "ANTIBIRDS PARASITIZE, FORAGING ARMY ANTS", Ecology, 86(3). 20005, pp.555-559.

[2] 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

[3] Steven Woodcock, 플로킹: 집단 행동을 흉내내는 간단한 기법 GAme Programmin Gems 1, pp.401-415, 2001.

[4] Steven Woodcock, 먹고 먹히는 플로킹: 포식자와 먹이, GAme Programmin Gems 2,이 pp.423-430, 2002.