인공 새 무리의 집단 행동 애니메이션

유 관 희†

요 약

본 논문에서 우리는 무리를 이루면서 살아가는 새의 집단 행동 애니메이션에 관해 논의한다. 먼저 개개의 새를 모델링하고 무리를 이뤄 살아가는 새들의 집단 행동을 위한 모델을 제시한다. 무리를 지어 생활하는 새들에게 나타나는 새들간의 충돌 방지와 모이기를 현실감 있게 표현하기 위해 우리는 새들의 집단 행동 욕구, 개개 새의 역할, 속도, 추진력, 기울어짐 요소 및 새 내부의 특성을 고려한다. 본 논문에서는 100마리의 집단 새 무리에 대한 제안된 모델의 모의 실험 결과를 제시한다.

A Behavioral Animation of Artificial Birds

Kwan-Hee Yoo†

ABSTRACT

In this paper, we explore a behavioral animation of artificial birds that have lived by doing an aggregate motion. We first model individual birds and then propose a behavioral model for an aggregate motion of a flock of birds. In order to represent realistically collision avoidance and flock centering among birds, which are necessary properties in a flock of birds, we consider motive of a flock of birds, and role, velocity, momentum, banking and internal characteristics of each bird. The paper presents the simulation result of the proposed model for a flock of 100 birds.

1. 개 요

새나 물고기는 여러 개체가 모여서 하나의 무리를 이루면서 살아간다. 이 무리는 개체 서로간의 상호 작용을 통하여 집단행동을 하게 된다. 또한 외부의 여러 가지 영향을 받으면서 그에 따른 행동을 하기도 한다. 이러한 집단 행동을 컴퓨터로 애니메이션(animation)하여 실세계를 가상의 컴퓨터 세계에 구현한다는 것은 매우 흥미로운 일이 될 것이다.

무리의 행동을 컴퓨터상에 애니메이션하기 위해 컴퓨터 상에 각 동물의 움직임을 개별적으로 지정해 주어야 하고 또한 동물의 움직일 수 있는 주변 환경을 기술하여야 한다. 그러나 집단으로 움직이는 동물의 수가 매우 많기 때문에 각 동물의 움직임을 일일이 기술하는 것은 불가능하다. 또한 각 동물들의 움직임을 기술한다고 할지라도 주변 환경이 바뀌면 처음부터 동일한 작업을 다시 해야 한다는 점에서 큰 어려움이 있다. 이러한 문제점을 해결하기 위해 컴퓨터 모의실험을 통한 애니메이션 기법이 도입되고 있다. 이 방법은 각 객체의 동작과 객체간의 동작을 모델링한 후 주변 환경 요소를 입력으로 받아 집단 동물의 움직임을 자연스럽게 애니메이션하는 기법이다[1,2,3,4].

집단 행동을 하는 동물의 무리를 제어하기 위해 객체의 자동 운동 제어 기법에 관한 연구가 Wilhelms[1]에 의해 이루어졌는데, 이 논문에서는 자동 경로 계획과 자극 반응을 미리 사용자가 주어야 하는 문제점이

† 정 회 원 : 충북대학교 컴퓨터교육과 교수
논문접수 : 1998년 11월 16일, 심사완료 : 1999년 1월 20일

있다. 그 후 Reynolds[2]는 스크립트 시스템(script system)에 기초한 모의실험 시스템을 개발하였다. 이 방법에서도 역시 사용자가 직접 모든 동물들의 움직임을 스크립트 파일에 등록하고 동물들의 움직임을 모의 실험한다. 따라서 이 연구에서도 각 동물이 움직임일 수 있는 모든 유형을 사용자가 구체적으로 제공해 주어야 한다는 문제점을 여전히 안고 있다.

Reynolds의 연구에서의 문제점인 사용자 입력 내용을 최대한 줄이기 위해 Takeuchi 등[4]은 모든 동물의 움직임 유형을 단 하나의 프로그램으로 처리할 수 있는 방법을 고안하였다. 그들은 모든 동물간에 가까워지고 멀어지는 특성을 유인 값(attractiveness)으로 나타내었다. 만약 동물간에 유인 값이 크면 서로 모이게 하고 유인 값이 작으면 서로 멀어지게 하는 모델로 무리를 이뤄 살아가는 동물의 행동을 제어하였다. 단순한 모델을 이용하여 좋은 성과를 거둔 연구이다. 그러나 동물의 행동은 Takeuchi 등이 제안한 유인 값 이외에도 각 동물의 특성 및 주변 환경에 의해 영향을 받으므로 이들 요소를 고려하여야 한다

본 논문에서 우리는 어떻게 하면 사용자의 부담을 줄이면서 실시간에 좀더 현실감 나는 새들의 집단 행동을 애니메이션할 수 있는지를 연구하였다. 우리는 사용자 입력 사항으로 주변 환경 요소를 고려하였으며, 이들 요소로 움직이고자 하는 방향을 나타내는 욕구와 움직임을 방해하는 장애물 정보를 주었다. 현실감 나는 새들의 움직임을 표현하기 위해 우리는 집단 무리에서의 각 새들의 역할, 새의 추진력, 새가 날아가는 속도 그리고 새가 날아갈 때 기울어지는 정도 등과 같은 새의 특성을 반영하여 새를 모델링하였다. 새의 역할은 새가 생성될 때 부여되며, 새는 각자의 역할에 따라 알맞은 일을 수행한다. 일반적으로 동물에게 계속적으로 같은 욕구를 주면 어느 정도 수준까지 그 욕구가 증가하게 된다. 반면 현재의 욕구와 다른 욕구를 줄 경우 현재의 욕구는 점점 감쇠 되고 새로운 욕구가 일어나게 된다. 이와 같은 특성을 우리는 새에게도 반영한다. 새로운 욕구에 의해 새의 날아가는 방향이 바뀔 수 있는데, 자연스러운 방향 전환을 위해 기울어짐 요소를 반영한다. 본 논문에서 우리는 이들 요소를 기반으로 무리를 이루면서 살아가는 새들에게 필수적으로 나타나는 충돌 방지, 흩어지기, 모이기 기법을 제시하고 구현하였으며, 또한 이들 기법을 통하여 새의 집단 행동 애니메이션을 수행하였다. 특히, 새의 내부 애니메이션을 위해 날개의 펄럭이는 동작을 추가하였다.

본 논문의 구성은 다음과 같다. 제2장에서는 무리를 이루며 살아가는 새의 특성을 반영하여 새를 모델링 한다. 새의 모델링을 기반으로 제3장에서는 새들간의 충돌방지, 흩어지기, 모이기 방법을 통한 집단 새의 애니메이션 기법을 제시한다. 제4장에서는 100마리의 집단 새를 이용하여 3장에서 제시한 모델링 기법을 모의실험한 결과를 논의한다. 마지막으로 본 논문에서 얻어진 연구 결과와 향후 연구 방향을 제시한다.

2. 새의 모델링

일반적으로 무리를 이루며 생활하는 객체들은 각자 주어진 역할을 수행하면서 집단 행동을 한다. 분명히, 무리를 지어 날아다니는 새도 각자 주어진 임무가 있을 것이다. 우리는 이러한 요소를 새들간의 집단 행동 모델링에 반영하기 위해 새를 지도자 새, 중재자 새와 보통 새로 구분한다. 지도자 새는 집단 행동을 이끄는 새로 날아가야 할 방향과 속도를 결정한다. 중재자 새는 지도자 새를 보좌하고 보통 새를 집단 무리에서 이탈하지 않도록 유도한다. 보통 새들을 공평하게 분산시켜 충돌을 방지하게 하고 서로 모이게 하기 위해 중재자 새들간에 거리 간격을 일정하게 유지하도록 한다.

무리를 지어 날아가는 새를 시뮬레이션하기 위해 꼭 필요한 요소가 새의 현재 위치와 날아가는 방향을 나타내는 정보이다. 계절이 바뀌어 이주할 때처럼 특별한 목적지가 있든지, 아니면 먹이를 찾아 다닐 때처럼 특별한 목적지가 없든지 간에 무리를 지어 움직이는 새들은 각자 날아가고자 하는 방향을 갖고 있다. 본 논문에서는 새들이 날아가고자 원하는 방향을 새의 욕구로 표현하고, 이러한 욕구를 6가지 - 위로 날기(FLY_UP), 아래로 날기(FLY_DOWN), 동쪽으로 날기(FLY_EAST), 서쪽으로 날기(FLY_WEST), 남쪽으로 날기(FLY_SOUTH), 북쪽으로 날기(FLY_NORTH) - 로 구분한다.

새가 원을 그리면서 날아갈 때나 방향을 바꾸기 위해 회전을 하면서 날아갈 때도 새는 분명 특정 방향으로 기울어져 있음을 볼 수 있다. 이러한 현상을 표현하기 위해 우리는 기울어짐 요소(banking)를 고려한다.

특히 새가 날아갈 때 기울어짐 요소를 적용하면 훨씬 자연스럽게 새의 움직임을 나타낼 수 있다. 우리는 새가 기울어지지 않고 날아갈 때 기울어짐 요소 값을 0으로 주었고, 동쪽 방향에 영향을 받으며 날아가는 경우 기울어짐 요소 값을 양의 실수로, 서쪽 방향에 영향을 받으면 날아가는 경우 기울어짐 요소 값을 음의 실수로 주었다.

모든 물체는 운동하던 방향을 유지하려는 추진력을 갖고 있다. 새처럼 날아다니는 동물들도 역시 추진력에 영향을 받는다. 즉 날아가는 방향을 갑자기 다른 곳으로 바꾸고자 하는 욕구가 있어도 곧바로 바뀐 방향으로 날아가지 못한다. 예를 들어, 동쪽으로 날아가던 새의 욕구를 남쪽으로 가라고 욕구를 바꿔주면 새의 방향이 일정 시간 동쪽을 유지하다가 동남쪽으로 그리고 남쪽으로 날아가게 된다. 이를 위해 본 논문에서는 현재의 욕구와 새로운 욕구가 같은 경우 새의 추진력을 증가 시키고, 다른 경우 현재의 욕구를 새로운 욕구로 서서히 변화 시킨 후 새의 추진력에 새로운 값을 할당하는 방식을 취한다.

새가 날아갈 때 단순히 날아간다는 동작 외에도 여러 가지 다양한 동작 머리를 움직인다, 날개를 펄럭인다, 다리를 움직인다 등 을 한다. 본 논문에서는 다양한 새의 동작 중에서 날개를 펄럭이는 동작을 추가하여 좀더 실감나는 애니메이션을 가능하도록 한다. 마지막으로 다른 새와의 충돌 검사를 효율적으로 하기 위해 새의 크기를 너비, 높이와 길이로 나타낸다.

지금까지 논의된 새의 모델링을 위해 사용된 주요 요소는 다음과 같다.

- int id; // 새의 식별자
- int role; // 무리에서는 새의 역할: "1"이면 새무리의 지도자 새, "2"이면 중재자 새, "3"이면 보통 새를 의미함
- double pos[3]; // 새의 위치를 나타냄
- double dir[3]; // 새의 날아가는 방향을 나타내는 벡터로 새의 욕구가 반영됨
- int motive; // 현재 날아가고 있는 새의 욕구 종류를 나타냄
- float momentum; // Motive에 따라 새들이 동작하고자 하는 정도를 나타냄
- float bank; // 새가 날아갈 때 기울어지는 정도를 나타내며 값의 범위는 1에서 1까지임
- float velocity; // 무리를 이루기 위해 새가 날아가는 속도를 나타냄
- int kind; // 새의 내부 애니메이션을 위한 현재의 날개의 동작을 나타냄
- float width; // 새의 너비를 나타냄
- float height; // 새의 높이를 나타냄
- float length; // 새의 길이를 나타냄

3. 새들간의 모델링

3.1 애니메이션

이제 2장에서 소개한 새의 모델링을 기반으로 우리가 새의 집단 행동 애니메이션을 어떻게 하였는지를 본 절에서 소개한다. 우선적으로 새 무리의 애니메이션을 위해 우리는 중재자 새와 지도자 새간의 거리를 조정한다. 중재자 새와 지도자 새간의 거리가 일정거리 (MIN_MIDDLE_DISTANCE) 이하이면 지도자 새와 중재자 새간의 거리를 벌려주고, 또 일정거리 (MAX_MIDDLE_LEADER) 이상 떨어져 있으면 서로 가까워 지도록 한다. 다음으로 모든 중재자 새들 사이의 상호작용을 고려한다. 중재자 새간의 거리를 항상 일정 거리 만큼 유지시켜 주기 위해 일정거리 보다 가까우면 벌려주고 멀면 모이게 한다. 그리고 모든 보통 새들에 대해 가장 가까이 존재하는 중재자 새를 선택하여 이 중재자 새와 충돌이 일어나는지를 검사하여 충돌이 발생하면 이를 해결하고 그렇지 않는 경우에 대해서는 보통 새가 중재자 새에 가까이 다가 가게 한다. 마지막으로 보통 새에 대해 이 보통 새를 관할하는 중재자 새 주변에 있는 몇 마리의 보통 새를 선택하여 이들 선택된 보통 새들간에 충돌이 일어나는지를 검사하고 충돌이 발생하면 충돌을 제어하고 그렇지 않으면 서로 가까워 지게 한다. 지금까지 논의한 집단 새 무리의 애니메이션에 관한 전체적인 알고리즘은 다음과 같다.

단계 1:
```
// 지도자 새와 중재자 새간, 중재자 새와 중재자
   // 새간 충돌 검사와 모이기 처리
for (k= 0; k < total_middle; k++) // total_middle은 중재
자 새의 수
{
```

```
    // 지도자 새와 k번째 중재자 새 간의 거리를 계산한 후
    // 너무 가까우면 일정한 거리만큼 벌리고,
    // 너무 멀면 모이기를 수행
    Distance = || 지도자 새의 위치-k번째 중재자 새의 위치||;
    if (distance < MIN_MIDDLE_DISTANCE)
        k번째 중재자 새가 지도자 새로부터 일정한 거리만
        큼 벌린다; // 상세내용은 3.2절 참조
    else
        k번째 중재자 새가 지도자 새에게 가까이 다가간
        다; // 상세 내용은 3.4절 참조
    for (k번째 중재자 새가 아닌 모든 중재자 새에 대해)
    {
     그 중재자 새를 m번째 중재자 새라 하자.
     Distance = || m번째 중재자 새의 위치-k번째 중재자
     새의 위치||;
     if (distance < MIN_MIDDLE_DISTANCE)
        k번째 중재자 새가 지도자 새로부터 일정한 거리만
        큼 벌린다; // 상세내용은 3.2절 참조
     else
        k번째 중재자 새가 지도자 새에게 가까이 다가간
        다; // 상세 내용은 3.4절 참조
    } // k번째 중재자 새가 아닌 모든 중재자 새에 대해
  }

단계 2: // 보통 새와 중재자 새간, 보통 새와 보통 새간의
        // 충돌 검사와 모이기 처리
 for (k=1; k < num_birds; k++) // num_birds는 새의 총수
 {
  // 모든 보통 새에 대해 이 보통 새에 가장 가까이 있는
  // 중재자 새를 선택하여 충돌 검사 후
  // 충돌이 발생하면 벌려주고, 그렇지 않으면 모이기를 수
     행한다.
  k번째 새가 보통 새이면 {
     k번째 보통 새와 가장 가까이 있는 중재자 새를 선택
     한다.
     그 중재자 새를 m번째 중재자 새라 하자.
     If (m번째 중재자 새와 k번째 보통 새간의 충돌검사)
     // 상세 내용은 3.3 절 참조
        k번째 새가 m번째 중재자 새에게 가까이 다가간
        다; // 상세 내용은 3.4절 참조
     k번째 새가 아닌 다른 보통 새를 임의로 몇 마리 선
     택한다.
     for (선택된 모든 보통 새에 대해)
     {
       선택된 하나의 보통 새를 m번째 보통 새라 하자.
       If (m번째 보통 새와 k번째 보통 새간의 충돌검사)
       // 상세 내용은 3.3 절 참조
          k번째 새가 m번째 새에게 가까이 다가간다; //
          상세 내용은 3.4절 참조
     } // m번째 보통 새와 선택된 m번째 보통 새와의 충
       돌검사 및 모이기
  }
 }

 단계 3: // 모든 새의 애니메이션
   모든 새를 애니메이션한다.
}
```

(그림 3.1) 무리 새의 애니메이션 처리를 위한 알고리즘

3.2 흩어지기

전체적으로 무리를 지어 이동하는 집단에서 지도자 새와 중재자 새는 일정한 거리를 두고 다른 보통 새들을 보호할 필요가 있으며 또한 보통 새들로부터 보호받을 수 있다. 따라서 지도자 새와 중재자 새 간 또는 중재자 새들간에 충돌은 일어나지 않지만 일정 거리보다 너무 가까이 있는 경우 이들 새들을 서로 멀어지게 할 필요가 있다. k번째 새를 m번째 새로부터 멀어지게 하기 위해 우리는 다음과 같은 방법을 사용한다.

k번째 새의 위치 = k번째 새의 위치+Dc x (m번째 새의 위치-k번째 새의 위치),

여기에서 Dc는 새들간에 얼마만큼 멀어지게 할 것인지를 나타내는 상수이다.

3.3 충돌 검사와 충돌 제어

새들이 상호동작을 하며 집단 행동을 할 때 서로간의 충돌이 일어나서는 안 된다. 우리는 두 새의 충돌 검사를 위해 먼저 새의 위치간의 거리를 계산한다. 새들이 날개를 가지고 있으므로 두 새의 충돌 여부는 두 새의 위치간의 거리와 새들의 날개 길이의 합에 의해 결정된다. 만약 두 새의 위치간의 거리가 날개 길이의 합보다 작으면 충돌이 일어난 것으로 가정한다. 충돌이 발생하는 경우 충돌을 피하기 위해 새가 기존 방향에 반대 방향으로 날아간 만큼을 계산하여 이를 새의 위치에 반영한다. 이를 위한 구체적인 방법은 다음과 같다.

```
// 두 새(A,B)의 위치간의 거리를 계산하고 충돌 검사를 하
// 고 충돌이 발생한 경우 이를 해결한다.
distance = || A새의 위치-B새의 위치||;
if (distance <= (A새의 날개의 길이+B 새의 날개의 길이)
{ // 충돌이 발생
   A 새의 방향 = A 새의 방향×-1.0;
   A 새의 위치 = A 새의 위치+A 새의 방향×A새의 속도;
}
```

(그림 3.2) 두 새의 위치 정보를 이용한 충돌 검사 및 제어

두 마리 새의 위치간 거리가 두 새의 날개의 합 보다 클 경우에도 새의 머리 부분과 꼬리 부분이 있기 때문에 충돌이 발생할 수 있다. 이를 효율적으로 검사하기 위해 먼저 충돌 가능성 여부를 계산하여야 하는데, 우리는 두 새의 위치간 거리가 두 새 날개의 합의

세배 이하이면 충돌 가능성이 있는 것으로 판단한다. 충돌 가능성이 있는 경우 먼저 새의 머리 부분에서 다른 새의 머리 부분과 꼬리 부분 각각에 대해 충돌 여부를 검사한다. 만약 충돌하면 새의 방향을 반대 방향으로 바꾸어 새의 위치를 계산함으로써 충돌을 피한다. 또 다른 경우로 새의 꼬리 부분과 다른 새와의 충돌 검사를 머리 부분과 꼬리 부분에 대해 각각 처리한다. 이 경우는 앞 경우와는 달리 새의 꼬리 부분에서 충돌이 발생하므로 현재 날아가고 있는 방향으로 새의 위치를 속도 만큼 전진시켜 주어 충돌을 피할 수 있다. 모든 경우에 대한 충돌 검사 방법과 충돌 해결 방법을 기술하면 다음과 같다.

```
// 두 새가 충돌 가능성이 있는 경우 먼저 새의 머리 부분
// 과 꼬리 부분에 대해 충돌 여부를 검사한다.
// 새의 위치, 길이, 방향 정보를 이용하여 새의 머리 위치
// 와 꼬리 위치를 구한다.
새의 머리위치 = 새의 위치+sqrt((새의길이×새의길이) / ||
                새의방향 ||)×새의방향; // 새의 머리 위치
새의 꼬리위치 = 새의 위치-sqrt((새의길이×새의길이) / ||
                새의방향 ||)×새의방향; // 새의 꼬리 위치
distance1 = || A새의 머리 위치-B 새의 머리 위치||;
distance2 = || A새의 머리 위치-B 새의 꼬리 위치||;
if (distance1 <= ((A새의 날개의 길이+B 새의 날개의 길
   이)×3.0) ||
   distance2 <= ((A새의 날개의 길이+B 새의 날개의 길
   이)×3.0))
{ // A의 머리 부분과 충돌이 발생하는 경우
   A 새의 방향 = A 새의 방향×-1.0;
   A 새의 위치 = A 새의 위치+새의 방향×새의 속도;
} else {
  distance1 = || A새의 꼬리 위치-B 새의 머리 위치||;
  distance2 = || A새의 꼬리 위치-B 새의 꼬리 위치||;
  if (distance1 <= ((A새의 날개의 길이+B 새의 날개의
                길이)×3.0) ||
     distance2 <= ((A새의 날개의 길이+B 새의 날개의
                길이)×3.0))
  { // A의 꼬리 부분과 충돌이 발생하는 경우
    A 새의 위치 = A 새의 위치+새의 방향×새의 속도;
  }
```

(그림 3.3) 두 새의 머리와 꼬리 위치 정보를 이용한 충돌 검사 및 제어

3.4 모이기

새들은 서로 모여서 무리를 이루려는 성질이 있다. 무리를 이루었을 때의 다양한 이점 때문이다. 적으로부터 자신을 보호하기가 용이하고, 먹이를 찾을 때도 혼자 있을 때보다 무리를 지어 다닐 때 훨씬 용이하기 때문이다. 또한 새들은 무리 중에서도 그 무리의 가운데로 몰리는 경향이 있다. 여기서 가운데라는 의미는 실제로 새 전체 무리의 가운데라기 보다는 자기 주변 새들의 가운데를 말한다. 자기 주변 새들의 가운데로 몰리다 보면 자연스레 커다란 하나의 무리가 형성된다.

무리에서 이탈한 새를 생각해 보자. 만약 무리에서 좀더 멀리 이탈된 새는 속도를 높여 빨리 날아가 무리에 접근하고자 할 것이고, 가까이에 있는 경우는 현재의 속도를 유지하면서 접근할 것이다. 우리는 이러한 특성을 반영하여 새들간의 모이기를 구현한다. 우리가 사용한 모이기 방법을 정리하면 다음과 같다.

```
// 두 새(A,B)의 위치간의 거리를 계산하여 속도를 조절한다.
D = || A새의 위치-B새의 위치||;
D 값에 따라 속도를 증감시켜준다. 특히 속도의 증감 시 최
고 속도 이상이거나 혹은 최저 속도 이하인 경우를 각각 최
고 속도와 최저 속도로 지정한다;
A새의 위치 = A새의 위치+A새의 속도×(B 새의 위치-A
새의 위치);
```

(그림 3.4) 두 새간 모이기 방법

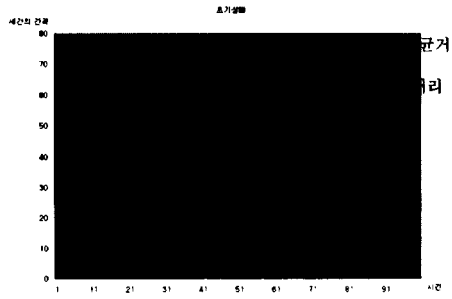
4. 구현 결과

우리는 PC Windows'95 환경에서 제2장에서 고려한 요소를 기반으로 3장에서 제시한 방법으로 새의 집단 행동을 모의 실험 하였다. 구현 도구로 Microsoft Visual C++[5]와 OpenGL[6]를 사용하였다. (그림 4.1) (a)가 공간상에 100마리의 새들로 구성된 초기 화면을 나타낸다. 메뉴에는 명령어로 위로 날기(FLY UP), 아래로 날기(FLY DOWN), 북쪽으로 날기(FLY NORTH), 남쪽으로 날기(FLY SOUTH), 서쪽으로 날기(FLY WEST)와 동쪽으로 날기(FLY EAST)를 제공한다. 이 명령어가 본 논문에서 고려한 욕구 요소로 작용한다. 그림에서 보이지는 않지만 각 새들은 초기 상태에서 시간이 지남에 따라 날개를 펄럭이고 있다.

초기 상태에서 지도자 새는 한 마리이며, 전체 100마리의 새들 중 6마리가 중재자 새이고 나머지 93마리가 보통 새이다. 위의 공간에서 몸의 색깔이 녹색인 새들이 중재자 새이다. 초기 상태에서 새들은 z축 방향을 향하고 있다. 초기 상태에서 새의 움직임을 모의 실험한 결과가 (그림 4.1) (b)에 나타나 있다. 세로 축

은 새들간에 떨어져 있는 평균 거리를 나타내고 가로축은 시간을 나타낸다. 그림에 나타난 실 선(solid line)은 보통 새와 모든 새간에 떨어져 있는 평균 거리를, 점 선(dotted line)은 지도자 새와 중재자 새간의 평균 거리를, 그리고 줄긋기 선(dashed line)은 중재자 새간의 평균 거리를 나타낸다. 그림에서 보는 바와 같이 초기에 100마리 새가 무작위로 생성되었기 때문에 초기에 서로 가까이 있다가 시간이 지남에 따라 점점 새들간에 일정한 간격을 유지함을 볼 수 있다.

(a)

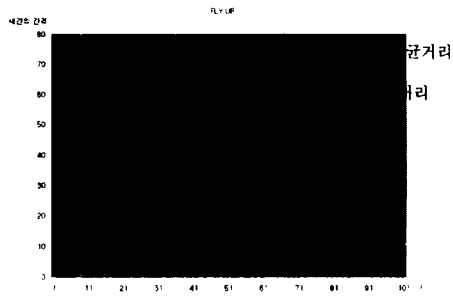


(b)

(그림 4.1) 초기 상태의 새의 집단 행동

초기 상태에서 사용자가 위로 날기(FLY UP) 욕구를 선택하면 (그림 4.2) (a)에서 보는 바와 같이 새들은 위를 향해 날아간다. 새들이 위쪽으로 날아가 공간 벽과 충돌하는 경우도 발생할 수 있는데, 이 경우 방향을 반대로 바꾸는 새들도 존재함을 볼 수 있고, 그럼에도 불구하고 욕구가 위쪽으로 날아가라는 성질 때문에 궁극적으로는 위벽에 모든 새들이 모임을 알 수 있다. 또한 새의 집단 행동 시 중재자 새간의 거리는 항상 일정한 거리를 유지하고 있고 보통 새들이 중재자 새들 주변에 고루 분포되어 있음도 알 수 있다. 사용자가 초기 상태에서 위로 날기 욕구를 선택할 시에 시간이 지남에 따라 지도자 새와 중재자 새간, 중재자 새와 중재자 새간, 보통 새와 모든 새간에 떨어져 있는 평균 거리의 변화가 (그림 4.2) (b)에 나타나 있다. 시간 축의 "0"에서 부터 "11"까지는 초기 상태를 나타내며, "11" 이후가 위로 날기를 선택하였을 때 새들간에 떨어져 있는 평균 거리를 나타낸다.

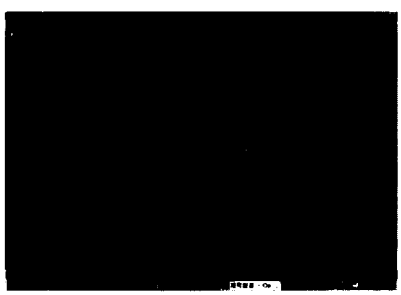
(a)



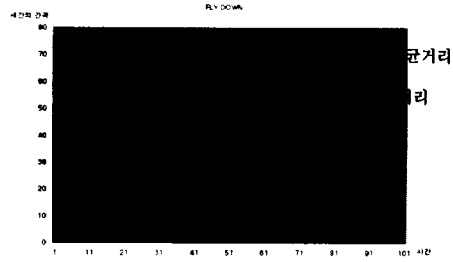
(b)

(그림 4.2) 초기 상태에서 위로 날기(FLY UP) 명령어를 선택했을 때 새의 집단 행동

위로 날기(FLY UP) 명령어 상태에서 아래로 날기(FLY DOWN) 명령어를 선택했을 때 새의 집단 행동 움직임은 (그림 4.3) (a)와 같고, 시간에 따른 모의실험 결과는 (그림 4.3) (b)와 같다



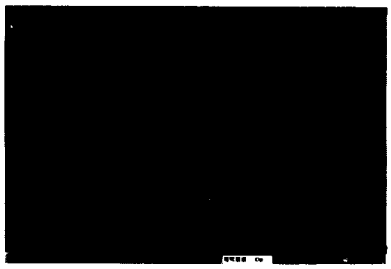
(a)



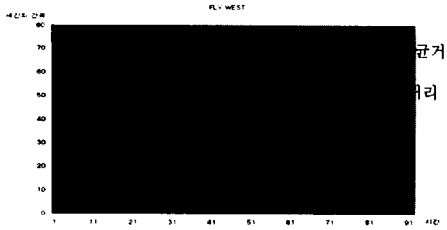
(b)

(그림 4.3) 위로 날기에서 아래로 날기를 선택했을 때 새의 집단 행동

(그림 4.4)와 (그림 4.5)는 각각 아래로 향하는 욕구에서 서쪽의 욕구를 선택했을 때와 서쪽의 욕구에서 동쪽의 욕구를 선택했을 때를 보여 주고 있다.



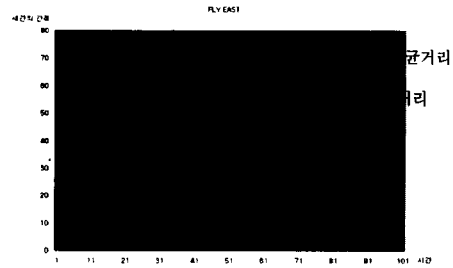
(a)



(b)

(그림 4.4) 아래로 날기 욕구에서 서쪽으로 날기를 선택했을 때의 새의 집단 행동

(a)



(b)

(그림 4.5) 서쪽으로 날기 욕구에서 동쪽으로 날기를 선택했을 때의 새의 집단 행동

5. 결론 및 향후 연구

본 논문에서는 사용자로부터 많은 정보를 받아 새 무리의 집단 행동 애니메이션을 처리했던 기존의 방법과는 달리 단지 날아가고자 하는 새의 욕구만을 입력받아 새의 집단 행동을 실시간에 현실감 있기 애니메이션할 수 있는 시스템을 구현하였다. 구현된 시스템에서는 무리에서 각 새의 역할, 추진력, 기울어짐, 모이는 특성, 충돌 방지 및 내부 애니메이션을 고려하였다. 향후 새의 나는 모습을 더욱 실감나게 하기 위해 집단에서의 주변 환경 요소로 공기역학을 고려할 예정이다. 일반적으로 날아가는 새의 방향과 속도는 바람이 어디서 어느 정도의 세기로 불어오는가에 따라 영향을 받을 것이기 때문이다. 또한 공기역학이 새의 내부 애니메이션에 어떠한 영향을 주는지를 분석하여 새의 움직임 모델링에 반영하고자 한다.

참 고 문 헌

[1] J. Wilhelms, "Towards automatic motion control," *IEEE Computer Graphics and Animation*, pp.11-22, 1987.

[2] C. W. Reynolds, "Flocks, herds, and schools: a distributed behavioral model," *Computer Graphics*, pp.25-34, 1987.

[3] M. Unuma and T. Takeuchi, "Generation of human motion with emotion," *Computer Animation '91*, pp.77-88, 1991.

[4] R. Takeuchi, M. Unuma, and K. Amakawa,

"Path planning and its application to human animation system," *Proceedings of Computer Animation '96*, pp.77-88, 1996.

[5] D.J. Kruglinski, *Inside Visual C++ Version 4.0*, Microsoft Press, 1996.

[6] R.S. Wright and M. Sweet, *OpenGL Superbible: The Complete Guide to OpenGL Programming for Windows NT and Windows 95*, Waite Group Press, 1996.

유 관 희

e-mail : khyoo@ghost.chungbuk.ac.kr

1985년 전북대학교 전산통계학과 졸업(학사)

1988년 한국과학기술원 전산학과 (공학석사)

1995년 한국과학기술원 전산학과 (공학박사)

1988년~1997년 (주)데이콤 종합연구소 선임연구원

1997년~현재 충북대학교 컴퓨터교육과 교수

관심분야 : 컴퓨터 그래픽스, 계산기하학, 3차원 네트워크 게임 등