한국컴퓨터정보학회 동계학술대회 논문집 제25권 제1호 (2017. 1)

Inverse Kinematics를 이용한 사족보행 게임 캐릭터 구현

정선효*, 강명주O
O*청강문화산업대학교 게임콘텐츠스쿨
e-mail: wjdtjsgysla@naver.com*, mjkkang@ck.ac.kr**

Development of Quadruped-Walk Game Character Using Inverse Kinematics

Sun-Hyo Jung*, Myung-Ju KangO
O*School of Game, Chungkang College of Cultural Industries

● 요 약 ●

본 논문에서는 Inverse Kinematics(IK) 방법을 이용하여 사족보행 게임 캐릭터의 이동 IK를 구현하였다. 게임 캐릭터의 이동 IK는 Unity의 NavMeshAgent를 사용하여 이동하는 사족보행 생물체가 회전 시 척추를 자연스럽게 꺾이도록 구현되었다. 본 논문에서는 게임캐릭터에 이동 IK를 적용했을 때와 적용하지 않았을 때를 비교하였으며, 이동 IK를 적용한 경우에 경로를 따라 이동하는 사족보행 캐릭터의 애니메이션이 자연스럽게 이루어짐을 확인하였다.

키워드: Unreal Engine4, Behavior Tree, FSM

I. Introduction

대형 보스몬스터가 등장하는 게임에서는 몬스터캐릭터가 화면을 차지하는 비율이 크다. 따라서 그 만큼 플레이어의 시선 집중도가 높아지며, 애니메이션의 디테일이나 자연스러움이 플레이어의 몰입도에 많은 영향을 미치게 된다.

본 논문에서는 애니메이션의 자연스러운 표현을 위해 Inverse Kinematics를 이용하여 대형 보스몬스터 캐릭터를 구현하였다.

II. Preliminaries

Forward Kinematics은 상위 Bone에서 하위 Bone으로 transform을 곱하여 각 Bone의 최종 위치를 구하는 것을 말한다[1]. 일반적으로 사용되는 키 프레임 애니메이션 방식은 이 Forward Kinematics의 일종이다.

Forward Kinematics는 애니메이션을 제작할 때 가장 보편적으로 사용되며 아티스트의 역량에 따라 섬세한 애니메이션이 가능하다는 장점이 있다. 그러나 정해진 애니메이션 외에는 출력할 수 없기 때문에 물체나 환경에 대한 유동적인 상호작용이 불가능하다는 단점이 있다. 따라서 게임에 등장하는 대형 보스몬스터는 Forward Kinematics만으로는 충분히 자연스러운 움직임을 표현할 수 없다[2].

따라서 본 논문에서는 위 문제를 해결하기 위해 Inverse Kinematics 방법을 이용하여 대형 보스몬스터 캐릭터를 구현하였다.

III. Inverse Kinematics

1. Inverse Kinematics

Inverse Kinematics는 1998년 GDC에서 처음 소개된 개념으로, 특정 위치를 기준으로 하위 Bone에서부터 상위 Bone으로 Bone의 transform을 정의하는 것을 말한다[3].

대부분의 게임캐릭터의 애니메이션은 스켈레톤에서 미리 정해진 값에 조인트를 변경하여 회전하는 것으로 처리된다. 자식조인트의 위치는 부모의 회전에 따라 변화하기 때문에 조인트의 종료점은 각도 및 포함된 개별 조인트의 상대적 위치에 의해 결정된다. 즉, 부모조인트의 움직임이 자식조인트의 위치와 회전에 영향을 준다[4]. 이와 같은 스켈레톤의 포징(Posing) 기술을 Forward Kinematics라고 하며, 일반적인 애니메이션은 Forward Kinematics를 통해 자연스러운 움직임을 구현할 수 있다. 그러나 게임캐릭터가 목표점을 터치하거나 평탄하지 않은 땅을 발로 붙이는 경우에는 자식조인트가 부모조인트의 위치와 회전 값에 영향을 주어야 한다. 이를 처리하는 방법이 Inverse Kinematics이다[1].

2. CCD IK

Inverse Kinematics를 구현하는 공식에는 여러 가지가 있다. 본 논문에서 사용한 공식은 CCD (cyclic coordinate descent)이다. CCD IK는 기본적으로 아래의 절차를 따른다.

(1) 연산은 가장 앞쪽의 본에서 시작한다.

(2) 현재 본에서 맨 끝 본의 꼭지점을 바라보는 Vertor와, Target을 향하는 Vector의 각도를 계산한다.

(3) 각도를 현재 본에 적용하고 다음 본으로 계산을 넘긴다.

(4) 일정 횟수만큼 본을 변경해가면서 반복한다.

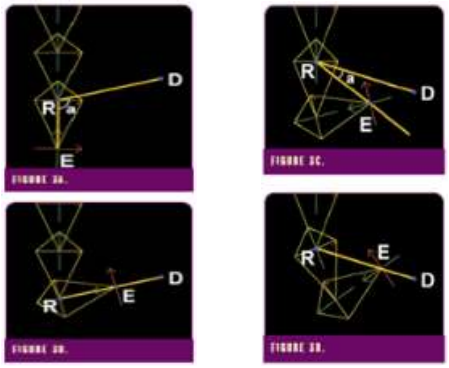

Fig. 1. CCD IK

IV. The Proposed Scheme

본 논문에서는 대형 사족보행 보스몬스터의 경로 이동에 대한 IK를 구현하였다.

1. 게임 소개

본 논문에서 구현한 게임은 근 미래의 도심에서 벌어지는 거대 보스와 전투하는 TPS 액션 게임으로, 3단계로 진화하는 보스가 도로와 골목, 지형에 따라 바뀌는 플레이 패턴이 특징이다.

보스 몬스터는 맵 전체를 이동하며 플레이어와 지속적으로 전투를 진행해야 하므로, 이동이나 상호작용이 자연스럽게 이루어져야 한다.

Fig. 2. Game Scene

2. Navigation Target Tracking

보스 몬스터는 Unity에서 제공하는 Navigation System인 NavMesh와 NavMeshAgent를 사용하여 이동한다. 이 보스 몬스터의 이동 IK를 제작하기 위해서는 보스 몬스터가 코너링 하여 몸통에 회전이 필요한 순간을 파악하여 IK의 Target을 설정해야 했다.

Target 설정 방법은 다음과 같다.

(1) 보스 몬스터의 이동 경로 선상의 일정 간격 앞에 Target을 위치시킨다.

(2) Target은 전방에 코너 지점이 존재했을 경우 그에 맞춰 이동하므로 보스 몬스터의 코너링 시 몸통 회전의 기준을 파악하여 처리한다.

3. Bone IK

NavTargeting단계에서 설정된 Target을 따라 척추가 휘어지기 위해서 CCD IK를 변형하여 사용하였다.

적용한 알고리즘은 다음과 같다.

(1) Bone 연산을 시도할 횟수, 각 Bone의 초기 Local 각도 등의 수치를 초기화한 뒤, 맨 끝의 Bone부터 연산을 시작한다.

(2) 현재 Bone에서 Bone의 꼬리를 향하는 벡터와 Target을 향하는 벡터를 구한 뒤, 두 벡터 사이의 각도를 구한다.

(3) 두 벡터의 외적 계산을 통해 각도의 방향을 구한 뒤 Bone의 회전 값에 각도를 적용한다.

(4) 각도가 적용된 회전 값이 해당 Bone의 각도제한에 걸린 경우 일정 각도로 제한하여 최종 적용될 각도를 구한다.

(5) 최종 각도를 월드 기준으로 Bone에 적용한다.

(6) 다음 Bone으로 동일한 방법으로 시도 횟수만큼 연산을 반복한다.

V. Results

a. 경로이동 IK가 적용된 몬스터 b. 경로 이동 IK가 적용되기 전 몬스터

Fig. 3. Results

[그림 3]의 a는 경로 이동 IK가 적용된 보스 몬스터의 모습이다. 그리고 [그림 3]의 b는 경로 이동 IK가 적용되기 전의 보스 몬스터의 모습이다. b의 경우 코너를 회전 시 몸통 전체가 한꺼번에 회전하기 때문에 부자연스러운 반면, a에서는 보스 몬스터의 이동 경로가 꺾여지면서 보스 몬스터가 바라봐야 하는 Target의 위치가 꺾이며, 보스 몬스터는 Target을 바라보게 되기 때문에 더 자연스러운 이동이 가능해졌다.

VI. Conclusions

본 논문에서는 Unity5의 Navigation System과 CCD IK를 이용한 사족보행 게임캐릭터를 구현하였으며, 게임캐릭터에 이동 IK를 적용했을 때와 적용하지 않았을 때를 비교하였다. 이동 IK를 적용한 경우가 경로를 따라 이동하는 사족보행 캐릭터의 애니메이션이 자연스럽게 이루어짐을 확인할 수 있었다.

REFERENCES

[1] http://t-robotics.blogspot.kr/2015/09/inverse-kinematics.html#.WE2jhLKLSUl

[2] KyeongWon Koo, Dynamic Animation IK, EA Sports MMA and its Use of Human IK, pp.21-22, Apr. 20, 2011

[3] 1998 GDC Math for Game Programmers: Inverse Kinematics Gino van den Bergen", November 1998: General Inverse Kinematics Solution". (2-5 pages)

[4] https://docs.unity3d.com/kr/current/Manual/InverseKinematics.html