2001年度 第14回 信號處理合同學術大會 論文集 第14卷 1號

휴먼 모션을 위한 3D 애니메이션 엔진 개발

남승우, 김도형, 이지형, 이인호

한국전자통신연구원 가상현실연구개발센터

Developing 3D Animation Engine for Human Motion

Nam Seungwoo, Kim Do Hyung, Lee Ji Hyung, Lee In Ho

Virtual Reality Research Center, ETRI

Virtual Reality Research Center, ETRI
161 Gajong, Yusong, Taejon 305-350, Korea
Phone: (042) 860-5314
Fax: (042) 860-1051
E-mail: {swnam, kdh99, ijihyung, leeinho }@etri.re.kr

1. 요약

하드웨어 성능 및 소프트웨어 기술에 힘입어 컴퓨터 애니메이션 기술은 발전을 거듭하고 있으며, 영화와 게임산업으로부터 군사훈련에 이르기까지 다양한 응용분야에서 이용되고 있다.[1][2]

개발된 3D 애니메이션 엔진은 구조화된 클래스들로 되어있기 때문에 클래스의 가감이 용이하다. 또한 모션 데이터 로더와 캐릭터 로더가 포함 되어 있고, 스킬(skill)간의 전이와 섞음이 가능하다. 그러므로 애니메이션 엔진을 이용하여 캡쳐된 모션데이터를 게임에 적용하기 쉬울 뿐만 아니라 다양한 가상환경에서 이용되리라 기대된다.

2. 서론

가상공간에서 캐릭터를 애니메이션하기 위해서 다양한 요구 사항이 있다. 캐릭터가 바닥에 빠지지 않도록 바닥 검사를 해야하고 다른 물체와 부딪치지 않도록 충돌 검사도 해야한다[3]. 또 다른 물체나 자기 몸에 부딪쳤을 경우 캐릭터의 반응을 운동학에 맞게 나타내야 한다[3][4]. 또한 현재의 모션에서 다른 모션으로 전이(transition)가 일어나면 모션을 부드럽게 연결하기 위해 모션데이터간 보간을 해야하고[5], 손을 흔드는 동작과 걸어 가는 동작이 있으면 두 동작을 섞어서 손을 흔들면서 걸어가는 모션 브랜딩(motion blending)을 해야한다.

본 논문에서는 바닥 검사, 충돌 검사, 모션 전이, 모션 브랜딩 등의 구현이 가능하도록 하면서, 클래스의 가감이 용이하게 하기 위해 구조화된 애니메이션 클래스 프레임웍을 제안하고, 개발의 연속성을 최대한 유지할 수 있도록 설계되었다. 따라서 본 3D 애니메이션 엔진을 이용하여 게임 캐릭터[6], 인터넷을 통한 서비스로서 가상 쇼핑몰의 점원[7], 영화 속에 캐릭터[8] 등과 같이 다양한 어플리케이션에 사용할 수 있으리라 기대된다.

캐릭터 제작에 쓰이는 대표적인 도구는 마야(Maya)와 3D Studio Max[9] 등이 있으며, 본 논문에서는 Max3.1 에서 자체 포맷으로 저장할 수 있는 플러그인을 제작하였고, 자체 포맷을 읽을 수 있는 로더를 엔진에 추가하므로써 캐릭터 제작과 모션 데이터를 적용하는데 용이하도록 하였다. 자체 포멧은 캐릭터의 geometry 정보를 담은 GME 포멧과 애니메이션 정보를 담은 AME가 있으며 max 에서 따로 저장한다.

개발한 3D 애니메이션 엔진은 구조화된 클래스로 구성되어 클래스의 가감이 용이하고, 클래스의 상속으로 코드의 재사용성을 높였다. 또한 동작 전이를 부드

럽게 하기 위해 skill 간의 보간과 동작 섞음을 위해 path 간의 교체를 할 수 있다. 마지막으로 Max3.1 플러그인을 통해 모션 캡쳐 데이터와 캐릭터 정보를 쉽게 엔진에 로딩할 수 있도록 하므로써 게임 뿐만 아니라 다양한 분야에 적용이 용이하도록 하였다.

3 장에서는 본론으로서 애니메이션 엔진에 사용된 클래스의 구조와 기능, 충돌 검사 및 모션 보간, 모션 섞음에 대해 설명하고 4 장에서는 결론을 맺는다.

3. 본론

3.1 클래스의 구조와 기능

각 클래스 구조는 그림 1 과 같다. CBase 클래스로부터 상속받은 클래스는 그림 1 에서 보는 바와 같이 CTickable, CActorInstance, CBehavior, CBoneHierarchy 등이며 CTickable 에서 상속받은 클래스는 CSpatialDataStructure 와 CActorInstance 이다. Vector 및 Matrix 연산을 다루는 클래스로는 CVector3D, CMatrix33D, CMatrix44D, CQuaternion 이고 CString 클래스와 CDebug 클래스는 독립적이다. Array 클래스로는 CBaseArray, CMatrix44Darray, CMatrix33Darray, Cvector3DArray, CQuaternionArray, CStringArray 등이 있으며, 각 변수 타입에 해당하는 Array 클래스가 존재한다.

표 1 주요 클래스의 기능

CBase	- name, alias - property - object comparing, assigning, creation
CWorld	- CActorInstacne, CSpatial-DataStructure handing - Dynamic and static object handling
CTickable	- message handling - status handling
CSpatialData-Structure	- scene manegement - collision detection - occlusion culling
CActorInstance	- animation instance handling - collision detection
CActorDefinition	- animation object template handling - run-time switching support
CBehavior	- behavior of animation object handling
CBoneHierarchy	- bone hierarchy handle
CBody	- geometry data of animation object handle
CSkill	- transform data of animation object handling - transition, blending support
CPath	- key frame data of animation object handling - key frame data modification support
CRunnable	- run interface
CDebug	- assertion, exception handling
CInterpolator	- interpolation method

표 1 에서는 주요 클래스의 기능을 나타내었다. 각 클래스의 기능은 크게 네부분으로 나눌 수 있다. 첫째, 물체의 애니메이션을 위한 모션파일을 핸들링하는 클래스이고 종류로는 CPath, CSkill 이다 둘째, 물체의 구조 및 geometry 를 핸들링 하는 클래스이고 CBoneHierarchy, CBody 이다. 셋째, 애니메이션 물체와 배경의 장면 변화를 핸들링하는 클래스이고 CSpatialDataStructure, CAactorInstance 등이 있다. 넷째, CWorld 가 첫째, 둘째, 세째를 통합하여 핸들링하는 클래스이다. 이와 같이 애니메이션 물체의 동작, 물체의 구조와 외형, viewing world 를 각각 핸들링하는 클래스를 따로 분리하였고 이를 통합하는 클래스를 두는 형태로 구조화 하였다. 또한 코드의 재사용성을 높이기 위해 클래스 상속을 사용였다. 각 클래스간의 객체들은 필요한 정보를 가지고 있는 다른 타입의 객체에 대한 레퍼런스(reference)를 가지고 있으며, 이를 통하여 데이터 교환을 위한 네트웍을 형성하고 있다. 그림 2 에서 각 클래스의 레퍼런스 다이어그램(Reference diagram)을 나타내었다.

3.2 충돌 검사 및 모션 보간

충돌 검사는 캐릭터와 캐릭터간의 충돌 검사와 캐릭터와 외부 환경과 충돌 검사, 캐릭터 자신의 신체 일부

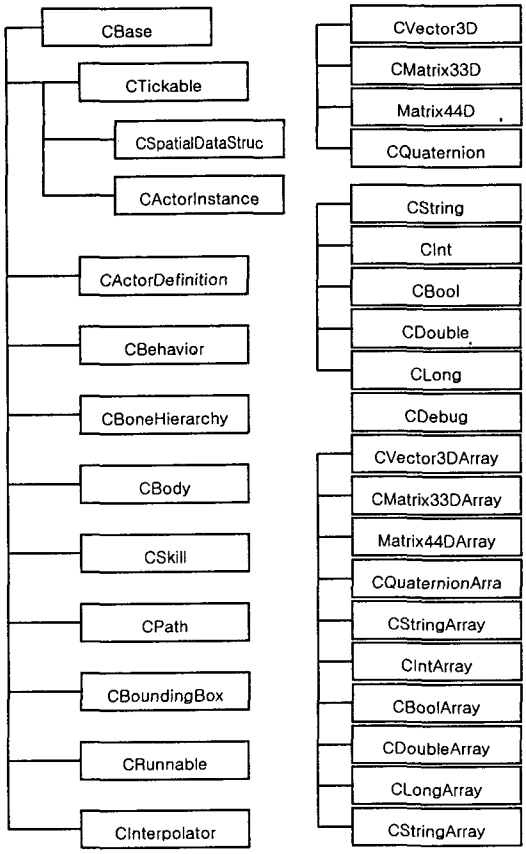


그림 1. 애니메이션 클래스의 구조

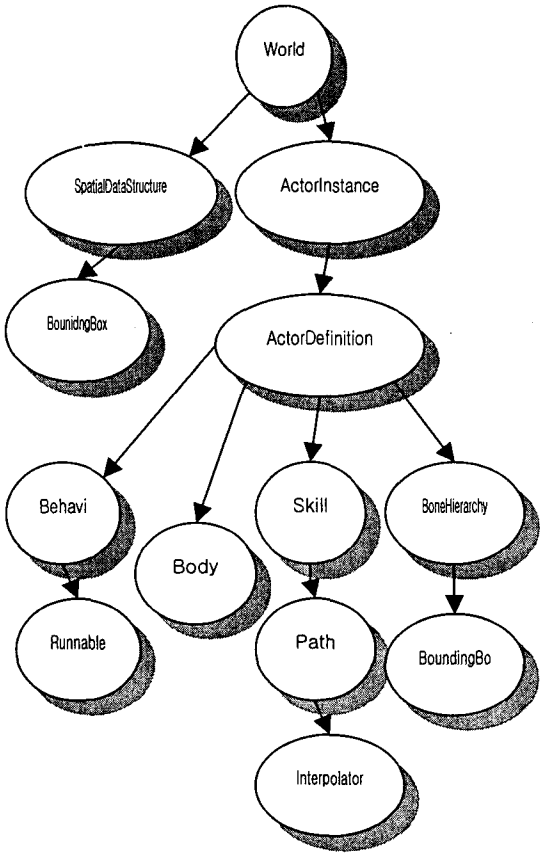
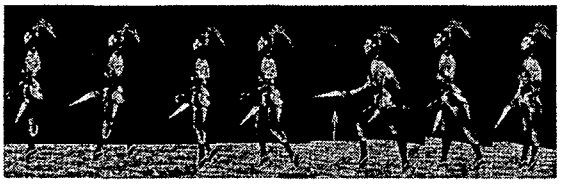


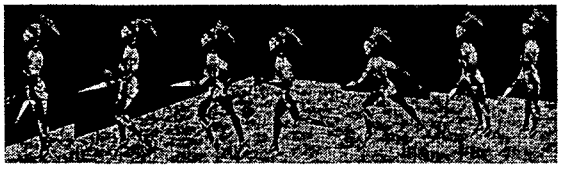
그림 2. 레퍼런스 다어어그램(Reference diagram)

경계구(bounding sphere)를 이용하는 것과 삼각형 단위의 충돌 검출 등의 방법이 있으며[3] 전자는 객처를 감싸는 경계구들의 반지름을 비교함으로써 객체와의 충돌 여부를 판정하는 방식이고, 후자는 매개변수 방정식을 이용해서 하나의 삼각형과 대상의 삼각형의 평면사이의 교점을 결정하고 그 점들이 대상 상각면 안에 존재하는지를 검사함으로써 충돌 여부를 판정한다. 본 엔진에서는 전자의 경계구를 경계상자(bounding box)를 이용하고 [10] 계산 방식은 후자의 매개변수 방정식을 이용하여 하나의 사각면과 직선과의 교점을 결정하고 그 점들이 대상의 사각면 상에 존재하는지 검사함으로써 충돌을 판정한다. 캐릭터의 분절마다. 최소의 외각 경계 상자를 가지고 있으며 캐릭터 자체의 최소의 외각 경계 상자도 가지고 있다. 캐릭터의 분절뿐만 아니라 캐릭터 자체의 경계 상자를 둔 이유는 LOD(level of detail)에 이용하기 위함이다.

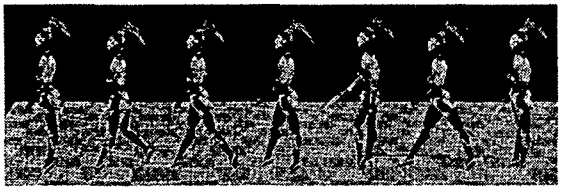
걷기에서 뛰기로의 전이를 하고자 할 때 걷는 동작을 끝내고 뛰는 동작을 시작하면 된다. 그런데 이런 경우 동작이 걷는 동작에서 뛰는 동작으로의 부드러운 전이가 일어나지 못하고 동작이 갑자기 전이 하는 현상이 일어난다. 본 엔진에서는 선형 보간을 통하여 걷는 동작에서 뛰는 동작으로의 부드러운 전이가 일어 나도록 구현하였다. 그림 3 에서는 걷는 동작(a), 뛰는 동작(b), 걷는 동작과 뛰는 동작을 섞어놓은 동작(c)를 보여주고 있다. 여기에서 사용된 캐릭터는 Dungeon Keeper2 에서 Mistress 라는 캐릭터 모델을 사용하였다[11]. 이를 이용하여 걷는 동작에서 뛰는 동작으로의 부드러운 전이를 위해서는 걷는 동작의 끝 부분과 뛰는 동작의 시작을 겹치도록 하면서 동작을 섞을 때 걷는 동작과 뛰는 동



(a) 걷는 동작



(b) 뛰는 동작



(c) 걷는 동작과 뛰는 동작의 보간

그림 3. 걷는 동작과 뛰는 동작의 보간(weight=0.5)

작의 웨이트(weight)를 조정함으로써 더욱 부드러운 동작 전이 효과를 얻을 수 있다. 웨이트에 대한 선형 보간이 이루어지며 캐릭터의 관절각은 오일러(Eular)대신에 쿼터니언(Quaternion)을 사용하고 보간은 구형 보간(Slerp) 방법을 사용한다.

모션 섞음은 스킬(skill)의 패스(path)를 교체함으로써 가능하다. 즉 걷는 동작과 손을 흔드는 동작이 있다면 걷는 동작의 팔에 해당하는 패스를 손을 흔드는 동작의 팔에 해당하는 패스와 교체함으로써 구현할 수 있다. 본 엔진에서는 다른 스킬간의 패스들의 교체를 위해 CPath 클래스와 CSkill 클래스를 두고서 모션 섞음이 가능하도록 구현되었다.

4. 결론

제안한 3D 애니메이션 엔진의 각 클래스의 구조와 기능 그리고 충돌 검사 및 모션 전이, 모션 섞음에 대하여 살펴 보았다. 서론에서 언급한 요구사항을 만족시키기 위하여 충돌 검사 및 모션 보간, 모션 섞음 등의 메소드를 제공하고, 클래스의 추가가 용이하도록 구조화 시켰고 코드의 재 사용성을 높이기 위해 클래스 상속을 사용하였다. 그러나 모션 전이에 있어서 전후 동작의 겹치는 정도와 다이나믹한 웨이트 조정은 등은 더욱 자연스러운 애니메이션을 위해 향후 과제로 남아 있다. 그리고 예를 들어 손을 뻗쳐서 물컵을 잡는 동작을 캐릭터가 취하고자 할 때 캐릭터의 크기에 따라 물컵까지 손이 닿을 수도 있고, 지나칠 수도 있고, 모자랄 수도 있다. 이런 경우 IK(Inverse Kinematics)를 이용하여 캐릭터의 크기에 상관없이 컵을 정확히 잡아오는 동작하는 온라인 리타겟팅(retargeting) 등이 향후 과제로 남아 있다.

REFERENCES

[1]. A. Pina, E. Cerezo, F. J. Seron, "Computer Animation: from Avaters to unrestricted Autonomous actor (A Survey on replocation, and Modelling Mechanisms)", *Comput. & Graphics*, vol. 24, pp. 297-311, 2000.

[2]. K. Kanev, T. Suguyamae, "Design and Simulation of Interactive 3D Computer Games", *Comput. & Graphics*, vol. 22, No. 2-3, pp. 281-300, 1998.

[3]. M. Deloura, Game Programming Gems, Charles river media, 2001

[4]. J. Park, D. S. Fussel, "Forward Dynamics based Realistic Animation of Rigid Bodies", *Comput. & Graphics*, vol. 21, No. 4, pp. 483-496, 1997.

[5]. C. F. Rose, B. Guenter, B. Bodenheimer, and M. F. Cohen, "Efficient Generation of Motion Transition using Spacetime Constraints", In *Proc. Of SIGGRAPH96*, pp. 147-154, 1996.

[6]. Http://www.sega.com

[7]. Http://www.idsoftware.com

[8]. Http://www.cityjoin.com

[9]. Http://www.discreet.com

[10]. Http://www.genesis3d.com

[11]. Http://www.dungeonkeeper2.com