3차원 애니메이션을 위한 타이밍 시스템 구현

송완서[1], 경민호[2], 석혜정[3]
아주대학교[123]
{lovedrop[1], kyung[2], dbdip[3]}@ajou.ac.kr

Timing System for 3D Animation Production

Wanseo Song[1], Min-Ho Kyung[2], Hae Jung Suk[3]
Ajou University[123]

요 약

3D 애니메이션 제작에서 동작의 타이밍(예를 들면 timing&spacing, slow-in, slow-out)은 연기의 의미와 느낌을 정확히 표현하기 위한 매우 중요한 요소 중의 하나이다. 따라서 이러한 타이밍의 편집은 애니메이션 작업에서 필수적이라고 할 수 있는데, 이를 기존의 3D 애니메이션 시스템에서 수행하기에는 기술적으로 많은 어려움이 있었다. 첫째로 타이밍의 편집은 시간 축 자체를 변형하는 문제이기 때문에 보간 곡선에 대한 재매개변수화가 필요한데, 이러한 기능은 기존 애니메이션 시스템에서 제공되지 않는다. 둘째로 타이밍 편집에는 종종 애니메이션 감독이 직접 참여하기도 하는데, 일반적으로 3D 애니메이션 시스템의 사용에 익숙하지 않기 때문에 원하는 결과를 직접 만들어 보기가 어려웠다. 본 논문에서는 이러한 문제들을 해결한 새로운 애니메이션 타이밍 시스템을 구현하였다. 이 시스템은 렌더링된 영상 파일들과 애니메이션 장면 파일을 입력 받아 사용자가 타이밍 편집을 하고, 그 결과를 애니메이션 장면 파일에 다시 기록하는 방식으로 구현된다. 타이밍 편집은 기존 셀 애니메이션 제작 방식과 유사한 방식으로 프레임을 삽입하거나 삭제하는 기능과 시간왜곡(time-warping) 그래프를 직접 조정하여 타이밍을 조정하는 기능을 제공한다. 전자는 제작 도구에 익숙하지 않은 감독이나 셀 애니메이션 작업자들이 직관적으로 사용할 수 있는 기능이고, 후자는 좀 더 세밀한 타이밍 조정을 위해 제공하는 기능이다. 사용자가 편집한 타이밍 결과는 각 동작변수의 보간 곡선을 재매개변수화하여 애니메이션 파일에 기록된다. 본 논문에서 구현한 시스템은 실제 애니메이션 제작에 보편적으로 사용되는 마야 애니메이션 파일을 지원하도록 구현되었다.

Keyword : Computer graphics, animation, user interface, reparameterization, animation timing

1. 서 론

애니메이션을 제작하는 데 있어, 동작의 타이밍 조정은 애니메이션의 완성도를 높이고, 애니메이션을 애니메이션답게 만드는 중요한 작업이다 [3]. 애니메이션은 영화와 달라서, 대상이 실제로 움직이는 그대로의 모습을 보여줄 때보다는 과장되고, 축약된 움직임에서 더 자연스러움을 느끼게 된다 [4]. 이런 과장되고 자연스러운 동작을 만들어내는 것이 바로 타이밍이다. 동작의 타이밍 조정은 다시말해 움직임 속도의 조정이라고 말할 수 있다. 예를 들어 *slow-in/slow-out*은 새로운 동작이 시작할 때 점진적으로 가속되고 끝날 때는 반대로 점진적으로 감속되게 하는 기법이다. 여기서 가속과 감속되는 정도를 조정하는 것이 타이밍에서 하게 되는 작업이다.

타이밍은 한편의 작품 속에서 전체적으로는 일관되게 관리되고, 부분적으로는 완성도 높게 만들어져야 한다. 이러한 이유로 타이밍 조정 작업에 감독이 직접 관여하기도 하는데, 일반적으로 3D 애니메이션 시스템의 사용에 익숙하지 않은 감독들은 원하는 타이밍을 애니메이터에게 구두로 지시한다. 그러나, 구두 지시의 부정확성은 감독과 애니메이터 간의 의사소통을 원활히 하는데 장애가 되어 결과적으로 작업

능률을 저하시킨다. 이런 문제점을 해결하는 것이 본 논문에서 제안한 타이밍 시스템의 개발 동기이다. 즉, 3D 애니메이션 시스템에 익숙하지 않은 사용자도 쉽게 타이밍을 조정하고 확인해 볼 수 있게 하는 것이다.

타이밍 조정은 능숙한 애니메이터에게 조차도 상당한 시간과 노력을 요구하는 작업이다. 하나의 캐릭터에는 동작을 제어하는 수십 가지 이상의 동작변수(위치, 방향, 관절각 등)가 사용되는데, 타이밍을 조정할 때 이 동작변수들의 변화가 정확히 동기화되어야 한다. 동기화의 의미는 타이밍의 조정에 의해 프레임 $i \rightarrow j$로 시간이 바뀌었을 때, 초기 프레임 i와 타이밍 조정후 프레임 j에서 모든 동작변수들의 값이 변하지 않도록 하는 것이다. 동기화가 되지 않을 경우 수정된 동작은 원래와는 다른 동작이 될 수 있다. 현재의 애니메이션 시스템에서는 수십 가지 동작변수의 키프레임 보간곡선들([2]참조)을 동기화된 형태로 수정하는 것을 모두 수작업으로 하여야 한다.

본 논문에서는 3D 애니메이션의 타이밍을 직관적인 인터페이스를 통해 효과적으로 조정할 수 있는 새로운 시스템을 제안한다(그림 1 참조). 이 시스템은 렌더링된 순차적인 애니메이션 영상 파일들과 애니메이션 장면 파일(scene file)을 입력 받아 사용자가 타이밍 편집을 할 수 있게 하여 주고, 그 결과를 다시 애니메이션 장면 파일에 기록한다. 효과적인 타이밍의 편집을 위해 두 가지의 방식의 사용자 인터페이스를 제공한다. 한 가지는 2D 셀 애니메이션 작업 방식에서 사용하는 타이밍 편집 방식이다. 이 방식은 간단히 프레임 단위의 이미지들을 복제하거나 삭제함으로써 타이밍을 편집한다. 다른 방식은 *시간왜곡(time-warping) 그래프*를 직접 조정하여 타이밍을 조정하는 것으로 첫 번째 방식으로 어려운 타이밍의 세부 조정을 가능하게 해 준다. 수정된 타이밍은 보간 곡선들을 재매개변수화(reparametrization)하는 과정을 거쳐 애니메이션 장면 파일에 기록된다.

본 논문의 구성은 2장에서는 관련된 연구 및 기존 시스템에 대한 분석을 하고, 3장에서는 타이밍 시스템의 사용자 인터페이스에 관한 설명을 한다. 4장에서는 타이밍 수정 결과를 반영하기 위한 보간곡선의 재매개변수화에 관해 설명하고 5장에서 구현된 결과

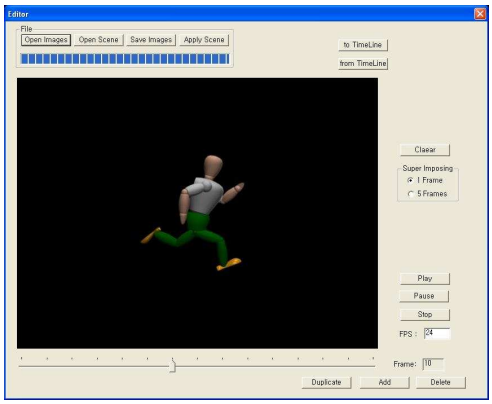

그림 1: 타이밍 시스템의 주 화면

를 보인다. 마지막으로 6장에서는 논문에서 제안된 타이밍 시스템에 대한 결론을 내리고, 앞으로의 연구 방향에 대해 제시한다.

2. 관련 연구

오늘날 3D 애니메이션 스튜디오에서 많이 쓰이는 3D 제작 시스템에는 기본적인 3차원 제작 환경과 함께 렌더링된 이미지로 애니메이션을 프리뷰해 볼 수 있는 기능이 통합되어 있다. 예를 들어 마야 애니메이션 시스템에 포함되어 있는 fcheck이라는 프로그램은 마야에서 렌더링된 순차적인 영상 파일들을 원하는 재생 프레임 수에 맞게 재생시켜 볼 수 있다. 하지만 전체적인 재생 프레임 수의 변화만으로는 다양한 타이밍 변화를 테스트 하는데 한계가 있다. 재생 기능에 덧붙여, 영상을 확인하면서 특정 프레임 영상에 텍스트와 간단한 그리기 기능으로 메모할 수 있는 기능을 제공하고 있다. 이러한 기능들은 감독과 애니메이터 간에 작업 내용에 관한 의사소통을 하는데 매우 유용하게 사용되고 있다.

3D 애니메이션 제작 시스템에서는 위에서 언급한 것 외에도 비선형 애니메이션 편집 툴이 있다. 예를 들어 마야의 Trax Editor는 사용자가 애니메이션과 사운드와의 동기화, 애니메이션 클립의 편집 기능을 제공한다. 애니메이션 클립을 가공하는 과정에서 타이밍의 조정이 가능하긴 하지만, 제한적인 기능만 제공하고 초보자가 쉽게 사용하기에는 직관성이 떨어진다.

그래픽스 분야에서 애니메이션 타이밍 편집에 관한 연구 결과는 많지 않았다. 직관적인 타이밍 적용에 관한 최근 연구로는 2004년 Terra와 Metoyer [7]가

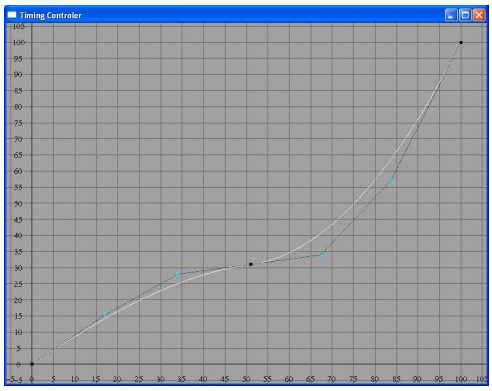

그림 2: 기존의 시간축과 새롭게 적용되길 원하는 시간축 간의 관계 그래프

키프레임 사이의 모션을 유지하면서, 그 시간 간격을 사용자가 손으로 시연한 타이밍으로 바꾸는 것을 제안하였다. 움직임의 시간 간격을 가장 직관적인 손을 이용해서 바꾼다는 장점에도 불구하고, 그 입력이 불완전하다는 점과 기존에 있던 모션과 동기화하는데 매번 성공할 수 없다는 단점을 갖고 있다.

REALVIZ사에서 개발한 ReTimer는 동영상을 빠르게 하거나 느리게 하여 다양한 시각 효과를 만들 수 있는 기능을 제공해 준다. ReTimer는 인접한 프레임 간의 픽셀 움직임을 분석하여 두 프레임 영상 간의 내삽된 중간 영상을 만들어 내고, 이 방법을 이용하여 연속적으로 시간축이 변형된 동영상을 생성해 낼 수가 있다. 따라서 ReTimer는 애니메이션의 최종 렌더링을 끝내고 후반 작업에서 타이밍을 수정할 때 효과적으로 사용될 수 있다. 하지만, ReTimer에 의해 만들어진 영상은 렌더링된 영상에 비해 품질이 떨어지고, 화면 변화가 큰 영상의 경우 내삽된 영상이 깨지는 단점이 있다.

본 논문에서 제안한 타이밍 시스템은 ReTimer와 유사한 기능을 하지만, 타이밍 수정 결과를 애니메이션 장면 파일에 직접 기록하기 때문에 이후에 고품질의 애니메이션 영상을 만들어 낼 수 있다는 장점을 가지고 있다. 또한 이미지 단위의 편집을 통해 타이밍을 조정할 수 셀 애니메이션 작업 방식을 도입함으로써 그래프 방식에 익숙하지 않은 사용자들에게 직관적인 인터페이스를 제공해 준다.

3. 사용자 인터페이스

타이밍 시스템의 인터페이스 방식은 앞에서 언급했다시피 두 가지가 제공된다. 순차적으로 렌더링된 영상을 가지고 프레임 단위로 타이밍을 조정하는 방식과, 시간왜곡 그래프를 직접 수정하여 정밀한 조정을 할 수 있는 방식이다. 시간왜곡 그래프는 가로축이 원래의 시간축이고 세로축이 변화된 시간축일 때, 원래의 시간이 새로운 시간으로 대응되는 관계를 보여주는 그래프이다. 즉, 그래프의 한 점 (t,t')은 원래 시간 t에서의 영상이 새로운 시간에서는 t'에 보여진다는 것을 의미한다. 시간왜곡 그래프는 초기에는 정비례 선분이 된다(그림 3).

3–1 프레임 단위의 타이밍 조정

렌더링된 애니메이션을 프레임 단위로 편집하여 타이밍을 조정하는 아이디어는 전통적인 셀 애니메이션 작업에서 비롯된다. 셀 애니메이션에서는 캐릭터가 그려진 셀들을 빠른 속도로 넘겨가면서 움직임과 타이밍이 적절한지 확인한다. 중간의 동작을 느리게 만들려면, 그 부근의 프레임들 사이에 추가로 여러 장의 셀들을 삽입하고, 빠르게 만들려면 반대로 그 부근의 셀들을 뽑아 내어 프레임 수를 줄여 준다.

이러한 작업 방식을 3D 애니메이션 제작에 적용하는 것은 여러 가지 장점을 가지고 있는데, 전통적인 셀 애니메이션 제작에서 확립된 여러가지 기법들을 3D 애니메이션에 적용하기가 쉬워진다. 실제로 미국 디즈니 사에서는 이러한 이유로 내부적으로 이러한 방식으로 타이밍 작업을 도와 주는 소프트웨어를 만들어서 3D 애니메이션 제작에 활용하고 있다고 알려져 있다. 두 번째로는 셀 애니메이션 제작에 익숙한, 또는 3D 소프트웨어 사용이 서투른 사용자들도 타이밍 효과를 쉽게 테스트해 볼 수 있다. 이러한 이유로 우리의 타이밍 시스템 인터페이스로 프레임 단위의 편집 방식을 채택한 것이다.

타이밍 시스템의 주화면은 그림 1과 같은 모습을 하고 있다. 중간의 프리뷰 창은 렌더링된 영상 파일들을 변화된 타이밍에 따라 순차적으로 재생해 준다. 사용자는 프리뷰 창 밑의 프레임 슬라이더를 움직여 원하는 프레임으로 이동해 갈 수 있고, 선택된 프레임에 대하여 프레임 반복, 삽입, 그리고 삭제를 통해 타이밍을 조정할 수 있다.

반복 선택된 프레임의 영상이 다음 프레임에도 추가

되어 반복되도록 한다. 애니메이션에서 이 기능은 중요한 순간에 일시적으로 정지해 있는 동작을 만들어 준다. 시간왜곡 그래프에서 보면 반복된 프레임은 그림 4에서와 같이 세로 축에 평행한 선분으로 나타난다.

삽입 선택된 프레임 이후에 프레임을 추가하여 느린 동작을 만들어 준다. 반복과 다른 점은 해당 프레임에서 동작이 정지하는 대신, 영상의 연속성을 유지하면서 시간을 늘려주는 기능을 한다. 이 기능의 결과를 시간왜곡 그래프에서 보면 그림 5과 같이 된다.

삭제 선택된 프레임을 제거함으로써 동작을 빠르게 만들어 준다. 시간왜곡 그래프에서는 가로축으로 평행한 선분으로 나타난다 (그림 6 참조).

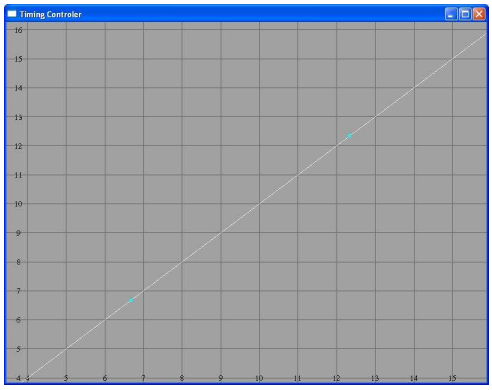

그림 3: 타이밍 조정 전의 시간왜곡 그래프

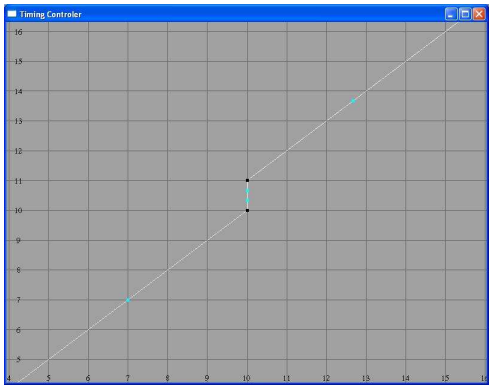

그림 4: 선택한 프레임을 반복한 후의 시간왜곡 그래프

3–2 시간왜곡 그래프를 이용한 타이밍 조정

앞에서 소개한 방식은 프레임 단위의 국지적인 타이밍 조정 작업에 용이한 반면 타이밍의 세밀한 조정은 불가능하다. 따라서 이러한 문제를 보완하기 위하여 시간왜곡 그래프의 직접 수정 기능도 제공한다.

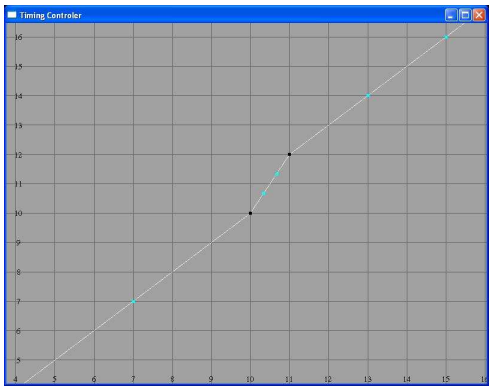

그림 5: 선택한 프레임 다음에 프레임을 삽입한 후의 시간왜곡 그래프

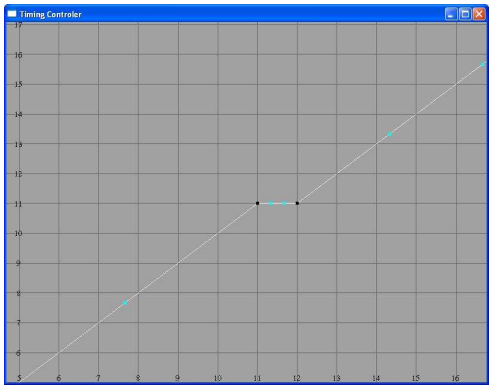

그림 6: 선택한 프레임을 삭제한 후의 시간왜곡 그래프

t 와 t' 사이의 관계를 유연하게 나타내기 위하여 3차 다항식을 이용한다:

$$t' = h(t) = at^3 + bt^2 + ct + d. \qquad (1)$$

이 곡선은 내부적으로 애니메이터들이 익숙한 3차 Bezier형태로 표현하였다. 따라서, 4개의 제어점(control point)들을 화면에 나타내고 제어점들을 조정하여 원하는 곡선의 모양을 만들도록 하였다. 하지만 시간왜곡 그래프에서 이 제어점들에는 몇가지 제한이 필요하다.

- 그래프에 극점이 생길 경우, 동작이 반대로 진행된다. 따라서 시간의 단방향성을 깨지 않기 위해서 그래프는 반드시 단조증가해야만 한다.
- 시작 프레임에서 끝 프레임까지만 동작 정보가 있으므로, 수평 축에 대한 양 끝점이 처음 시작 프레임과 끝 프레임을 벗어날 수 없다.

초기에는 전체 그래프가 직선 형태의 Bezier 곡선 하

나로 주어지게 되고, 상황에 따라 더 복잡한 곡선 모양이 필요할 경우 Bezier 곡선을 분할할 수 있도록 하였다. 줌 기능이나, 좌표축의 이동과 같은 인터페이스는 마야에서 제공하는 동작 곡선의 그래프 에디터와 같다 [5].

3–2–1 경과점

애니메이션의 동작 중에는 그 전체 움직임을 특징있게 만들어주는 반드시 거쳐야할 중요한 자세가 있다. 이것을 *경과점(passing position)*이라고 하는데, 타이밍을 조정하는 과정에서 이 경과점이 소실될 수 있다. 이 것은 프레임 값이 항상 자연수가 되어야 하기 때문이다. 예를 들어 현재 10번 프레임이 타이밍 조정 후에 프레임 값이 12.3이 되었다면 이 프레임은 렌더링되지 않고 12번 프레임에서 13번 프레임으로 건너뛰게 된다. 만일 이 프레임의 영상이 경과점이었다면 최종 렌더링에서 의도하지 않은 결과를 낳게 된다. 이를 방지하기 위해서 사용자가 이 경과점을 지정할 수 있게 하였다. 지정된 경과점은 시간왜곡 그래프에서 가로축의 한 프레임이 되고, 그래프가 그 프레임을 지나갈 때 반드시 세로축의 정수 값에 대응하는 곳을 지나가도록 만들었다(그림 7).

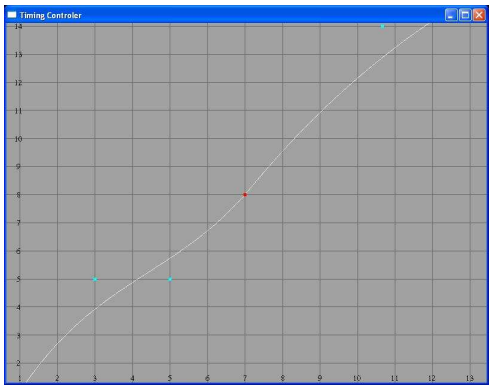

그림 7: 경과점을 설정한 경우

4. 보간곡선의 재매개변수화

장면 파일에는 각 동작변수의 곡선이 여러개의 키프레임으로 나뉘어져 있고, 키프레임과 키프레임 사이는 Bezier나 spline곡선으로 보간되어 있다. 본 논문에서는 보간 곡선을 마야에서 기본적으로 사용하는 3차 Bezier로 가정한다.

각각의 Bezier 보간 곡선은 다음과 같이 표현된다.

$$v = f(u), \quad t = g(u). \tag{2}$$

v는 한 동작변수를 나타내고, t는 프레임 값을 나타낸다. 곡선 매개변수 u에 대한 v와 t의 값은 Bezier 함수 f와 g로 정의된다.

타이밍 조정은 결국 시간축을 t에서 $t' = h(t)$로 변환하는 것으므로, 보간 곡선이 (t, v)에서 (t', v)으로 바뀌는 것이라고 볼 수 있다. 이를 수식으로 표현하면, t를 $g(u)$로 치환하여

$$t' = h(g(u)) = (h \cdot g)(u) = g'(u) \tag{3}$$

와 같이 나타낼 수 있다. 여기서 h은 최대 3차 함수이므로, g'은 u에 대한 최대 6차 다항식이 된다. 그런데, 대부분의 애니메이션 시스템은 이처럼 높은 차수의 보간 곡선을 지원하지 않기 때문에 g'을 그대로 쓸 수 없다. 따라서, 이를 다시 3차 Bezier함수로 근사시키는 과정이 필요하다.

3차 Bezier함수로 근사하는 과정은 다음과 같은 단계로 이루어진다.

1. u를 $[0,1]$ 구간에서 일정한 간격으로 n개의 점을 샘플링한다.
2. 최소자승법(least-squares method)를 이용하여 샘플링된 점들을 3차 Bezier 함수로 근사시킨다.
3. 근사 오차를 계산하여 임계치를 넘을 경우 u의 구간을 둘로 나누어 각각에 대해 근사를 반복한다. 임계치를 넘지 않을 경우 Bezier 함수를 출력한다.

여기서 근사 오차의 임계치를 적절히 설정하는 것이 중요하다. 지나치게 작게 설정하면 정확한 Bezier함수를 얻을 있지만, 너무 많은 Bezier 곡선들이 얻어진다. 보간 곡선이 너무 많은 Bezier 곡선들로 연결되면 이후에 애니메이터가 추가로 작업할 때 많은 번거로움이 발생한다. 임계치가 너무 크게 설정되면 근사 정밀도가 떨어지고 이 것은 동작변수들간의 동기화가 잘 이루어지지 못하는 결과를 낳는다. 최악의 경우 원래의 동작과 완전히 다른 동작이 나올 수도 있다. 임

계치를 어떻게 설정하는 것이 최적인지는 좀 더 연구가 필요하고, 본 타이밍 시스템에서는 일단 애니메이터가 임의로 줄 수 있도록 하였다.

5. 결과

타이밍 시스템에서 프리뷰창에는 이미 렌더링된 영상 파일들을 이용해 타임 변환에 맞추어 순차적으로 보여주기 때문에, 시간이 길어지는 경우 부드럽게 재생되지 않는다. 하지만, 대략적인 타이밍의 변화는 사용자가 충분히 확인할 수 있었다. 프레임 단위의 삽입, 삭제를 이용한 타이밍 조정 방식은 실제 애니메이션 제작팀에서 테스트해 본 결과 매우 긍정적인 반응을 얻었다. 3D 애니메이션 제작 시스템에 익숙하지 않은 사용자들 뿐만 아니라, 전문 애니메이터들도 이 간단한 인터페이스가 다양한 타이밍 효과를 쉽고 빠르게 테스트 해 볼 수 있어 작업 효율을 많이 높여 준다는 반응을 보였다.

원하는 타이밍 효과를 프레임 단위 인터페이스를 이용해 테스트해 본 후에, 시간왜곡 그래프를 이용해 정밀한 조정을 함으로써 부드러운 동작의 애니메이션을 최종적으로 만들어 낼 수 있었다. 그림 8는 시간왜곡 그래프를 수정한 것이고, 그림 9는 그 결과를 보여주고 있다. 위의 2, 3, 4번 이미지는 아래에서 각각 1, 3, 5번 이미지와 동일하다. 아래 2, 4번 이미지는 새로 생성된 이미지로 그 중간을 연결한다.

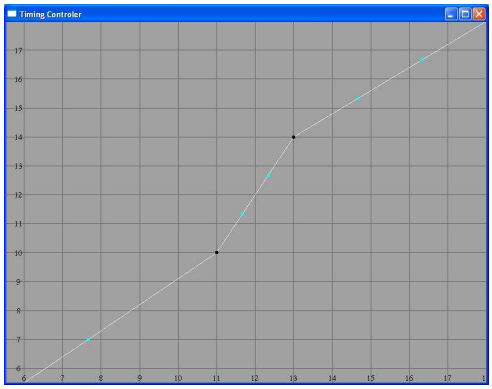

그림 8: 편집 전, 후의 시간왜곡 그래프

6. 결론

본 논문에서는 3D 애니메이션 제작에 필요한 새로운 타이밍 시스템을 개발하고 그 결과를 보였다. 이 타이밍 시스템은 기존 애니메이션 제작 시스템에서 많은 시간과 노력이 필요했던 타이밍 효과의 조정을 손쉽게 할 수 있는 두 가지 인터페이스를 제공하였다. 하나는 프레임 단위로 삽입, 삭제를 하여 타이밍 효과를 쉽게 테스트해 볼 수 있도록 하였고, 다른 하나는 세밀하고 정확한 타이밍 조정을 위해 시간왜곡 그래프 상에서 조정할 수 있게 하였다. 그리고, 이러한 결과를 애니메이터가 쉽게 작업에 반영할 수 있도록 애니메이션 장면 파일을 타이밍 수정에 맞도록 자동적으로 수정할 수 있도록 하였다. 이 과정은 기술적으로 곡선의 재매개변수화를 계산하여 구현하였다.

본 논문에서 개발한 타이밍 시스템은 애니메이션 감독과 애니메이터간의 의사소통 도구로써 편리하게 활용될 수 있다. 특히, 감독과 애니메이터가 지리적으로 떨어져 있는 경우에도 애니메이터가 렌더링된 애니메이션 동영상을 인터넷을 통해 감독에게 전송하기만하면, 감독은 자신의 컴퓨터에서 타이밍 시스템을 이용하여 손쉽게 타이밍을 수정하고 새로운 타이밍 효과를 테스트해 볼 수 있다. 그리고, 그 결과는 시간왜곡 그래프만을 애니메이터에게 전송함으로써 정확하고 빠르게 전달될 수 있다. 시간왜곡 그래프를 저장하고 전송하는 것은 아직 구현되어 있지 않지만, 간단히 추가될 수 있는 기능이다.

앞으로 추가로 개발이 필요한 부분으로 프리뷰 영상을 실시간으로 렌더링하여 프리뷰 창에 보여주는 기능이 있다. 프리뷰 창에서 보여지는 영상은 미리 렌더링된 순차적인 영상들이기 때문에 중간 프레임의 영상은 보여줄 수가 없다. 이러한 중간 프레임 영상을 만들어 내는 방법으로는 3D 렌더러를 내부적으로 불러서 렌더링하는 방법과, 하드웨어 가속기를 이용하여 낮은 품질이지만 실시간으로 보여 주는 방법, 그리고 이미지 처리 기술을 이용하여 중간 영상을 내삽하여 만들어 내는 방법 등이 있을 수 있다. 우리는 이 중에서 하드웨어 가속기를 이용하는 방법을 고려하고 있다. 또 다른 추가 기능으로 네트워킹 기능을 들 수 있는데, 두 타이밍 시스템 간에 네트워킹을 가능하게 하면 애니메이션 감독과 애니메이터 간에 더욱 원활한 원거리 피드백이 가능할 것이다.

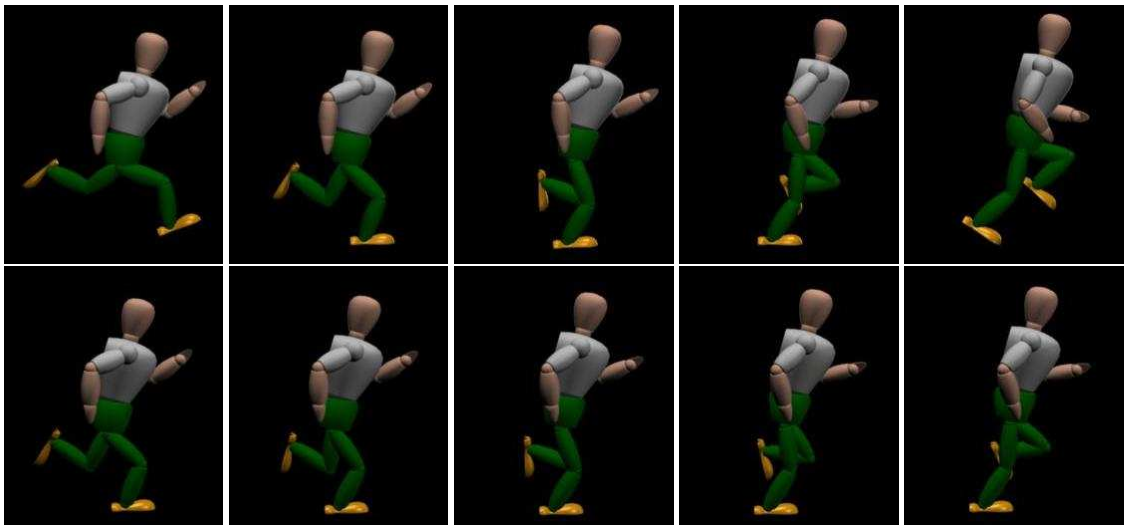

그림 9: 편집하기 전, 후의 프레임별 이미지
(위:편집 전의 10∼14 프레임, 아래:편집 후의 10∼14 프레임)

A 3차원 애니메이션 장면 파일 내 모션 정보

애니메이션 장면 파일의 모션 정보는 물체의 애니메이션 정보를 위해 나눈 각 채널 별로 저장되있다. 그리고 이 각각의 채널은 키프레임 단위로 정보가 들어있다. 그래서 키프레임 사이의 프레임은 두 개의 키프레임 사이를 곡선으로 연결시켜 나오는 값을 이용한다. 이와 같이 두 개의 키프레임 사이를 연결하는 하나의 곡선을 전체 곡선에 대한 조각 곡선이라 한다. 그리고 이 조각 곡선은 각각 제한된 형태의 3차원 Bezier 곡선 두 개로 표현되있다 [2]. 시간에 대한 곡선과 값에 대한 곡선이다. 조각 곡선은 두 키프레임 사이를 잇는 역할을 하므로 보간 곡선이라고도 한다.

키프레임에 대한 주요 정보는 키프레임의 인덱스, 시간, 값, 들어오는 접선 기울기, 나가는 접선 기울기 등이다.

참고 문헌

[1] Thomas W.Sederberg : An Introduction to Bezier Curves, *tom.cs.byu.edu* January 6, 2003.

[2] class MFnAnimCurve documentation *maya 6.0 unlimited*, Alias Systems, a division of Silicon Graphics Limited, 2004.

[3] Lasseter J.: Principles of traditional animation applied to 3d computer animation. In *Proceeding of the 14th annual conference on Computer graphics and interactive techniques* 1987. ACM Press.pp.35-44

[4] Christoph Bregler, Lorie Loeb, Erika Chuang, and Hrishi Deshpande, Turning to the Masters: Motion Capturing Cartoons, *SIGGRAPH* 2002

[5] Maya Help *maya 6.0 unlimited*, Alias Systems, a division of Silicon Graphics Limited, 2004

[6] William H. Press, Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery : Singular Value Decomposition *Numerical Recipes in C++*, CAMBRIDGE, 2002, 62 - 74

[7] S.C.L. Terra and R.A. Metoyer, Performance Timing for Keyframe Animation, *The Eurographics Association* 2004