스트리밍 프로세스를 적절히 이용하라!

2 Ratchet & Clank의 사례 … 프로토타입 제작한 다양한 시도 성공

게임 애니메이션을 위해 100명이 넘는 다양한 캐릭터를 만들어야 할 때 처음부터 모든 것을 해내기는 불가능하다. 또한 언제나 주어진 시간은 부족하다. 이런 상황을 극복하기 위해 예술적 표현 기술인 '마야(Maya)'를 통해 시간을 효과적으로 이용할 수 있는 몇 가지 방법을 소개한다. 시험 방식과 MEL 단순화 방법, 실시간 애니메이션 과정에 관한 기술적 세부사항이다. 이런 방법을 통해 제작팀은 제작시간을 줄이고 예술적 목표를 달성할 수 있을 것이다.

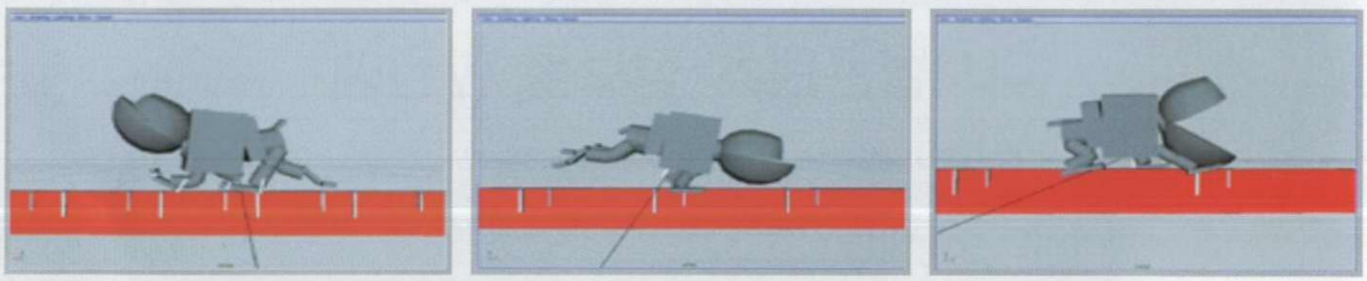
「편집자 주」

캐릭터 제작자로서 주인공이 둘, 10여명의 적, 20여명의 NPC 및 100여명 이상의 인물이 등장하는 캐릭터를 제작해야 했을 때는 최고의 프로젝트라고 느꼈다. 하지만 Ratchet & Clank의 비전은 이 디지털 배우들이 단순히 자동화된 반복 동작을 구현하는 것 이상의 작품이어야 하는 것이었다. 각각의 캐릭터들을 하나의 중간 과정으로 간주했으며, 이를 통해 플레이어에게 다가가고 그들을 새로운 세계로 깊숙이 끌어들이려는 것이 궁극적 목적이었다. 결국 이 캐릭터들이 물리적으로 환경과 조화를 이루어야 하고, 감정적으로 환경과 하나가 돼야 하며 내용과 하나가 돼야 하는 것을 의미한다.

프로토타입을 이용한 시험

캐릭터를 게임 플레이 및 환경과 밀접하게 연결시키기 위해 저해상도의 캐릭터들과 그에 대한 각각의 애니메이션 프로토타입을 제작했다. 프로토타입을 새로운 레벨에 적용시켜서 애니메이션, 프로그래밍 및 디자인의 잠재적 문제점들을 찾아내는 데 사용했다. 제작 과정에서 프로토타입은 캐릭터의 동작 세트를 정교하게 다듬는 수단으로 사용됐다. 이런 세밀화 과정은 고해상도 캐릭터를 구현해 애니메이션을 제작하기 전에 실행 불가능한 아이디어들을 선별해 제거하는 데 핵심적 역할을 했다.

일반적으로 사용된 프로토타입은 형태보다는 기능이 강조됐다. 비록 미적인 기준을 낮게 설정했더라도 이 프로토타입 모델들은 유효한 시험 케이스로서 기능을 충분히 실행할 수 있을 만큼 정확하게 구현되고 제작돼야 했다. 애니메이션 제작자의 입장에서 이 프로토타입은 정확한 높이만큼 점프할 수 있어야 하고, 디자인 사양을 충족시킬 뿐 아니라 적합한 속도로 달릴 수 있어야 했다.



〈그림1〉 개 캐릭터의 프로토타입을 사용하여 최종 캐릭터의 걷기, 뛰기 및 공격 등 애니메이션이 사전에 설정되었다.

프로토타입은 캐릭터 디자인 스케치를 가이드로 제작됐다. 이들 프로토타입의 캐릭터는 〈그림1〉처럼 기본적인 물체의 형상으로 제작되며 향후 구현될 모습과 대략적으로 닮았다. 이들 모델들은 구축이 쉽기 때문에 어느 애니메이션 제작자라도 모델링 경험과는 상관없이 구축을 도와줄 수 있다. 다만 캐릭터의 크기, 비율 및 자세 표현은 정확성이 요구된다.

대부분의 경우 원형 모델들은 극히 단순한 뼈대구조를 가지고 있다. 모든 기하학적 요소들은 특별한 변형 없이 하나의 뼈대로 지정돼 있다. 단순하기 때문에 모양은 둔하게 보일 수 있지만 실제에 있어서 애니메이션 제작자의 동작 세트 시험에 필요한 모든 유연성을 제공했다.

원형 캐릭터 제작은 연필 스케치 테스트와 유사하다. 애니메이션 제작자들에게 디자이너들이 승인한 동작 세트가 주어지지만 애니메이션은 가장 기초적인 형태로만 만들어지면 된다. 다듬고 겹치는 작업이 필요하지 않기 때문에 한번의 통과로 충분하다.

정확성이 필요한 부분은 타이밍, 치수 및 다른 캐릭터들과의 상호작용이다. 이들은 게임 플레이에 직접적으로 큰 영향을 미치기 때문에 이 사항들은 새로운 캐릭터의 동작을 정확하게 테스트하는 데 있어서 중요한 사항으로 간주된다.

타이밍은 애니메이션의 가독성과 게임 플레이에 중요한 효과를 가지고 있다. 타이밍이 제대로 설정되지 않을 경우 모호하게 보일 수 있다. 공격 동작에서 너무 느리거나 빠른 경우 적절한 효과를 거둘 수 없다. 따라서 구현하고자 하는 실시간 세계의 짧은 사이클 내에서 단 몇 가지 프레임에 대한 적절한 강조로 애니메이션이 성립되거나 프레임에 대한 미흡한 처리로 애니메이션이 실패할 수 있다. 원형 단계에서 타이밍을 시험하고 정밀하게 조정함으로써 최종 캐릭터의 애니메이션 재 작업을 피할 수 있다.

이밖에 프로토타입의 캐릭터들이 디자인 치수와 동일한지를 확인하는 작업이 중요하다. 예를 들면 4 미터의 거리에서 적이 공격하도록 디자인돼 있다면, 애니메이션 제작자는 프로토타입이 정확히 이 동작을 취하는지 확인해야 한다. 그렇게 하면 디자이너들은 적이 정확한 속도로 이동하는지, 적절한 난이도로 조정되는지, 또는 주요 캐릭터와의 척도가 적절한 지를 정확히 알 수 있게 된다.

프로토타입 제작은 또한 캐릭터의 행동과 상호작용을 사전에 시험할 수 있는 수단이 됐다. 원형 모델을 통해 Ratchet 또는 Clank 환경, 다른 캐릭터 등과의 상호작용을 초기에 파악할 수 있었다. 아티스트, 프로그래머와 디자이너들은 미리 보기를 통해 기술적 이용가능성과 게임 플레이의 가치 측면에서 캐릭터들의 행동을 알 수 있었다.


〈그림2〉 최종적인 개 캐릭터 모델

미리 보기 프로세스는 애니메이션 제작자뿐 아니라 디자인 및 프로그래밍 담당자들에게도 매우 유용했다는 판단이다. 이를 통해 프로그래머들은 게임 플레이 코딩을 시작할 수 있었고, 디자이너들은 각 단계에서 시험, 조정 및 변경을 요구할 수 있었으며, 정교한 조정이 가능했다.

애니메이션 제작자들은 미리 프로토타입을 만들어 시간과 노력을 절약할 수 있었다. 프로토타입을 만들어 사전에 시험해 보지 않았다면 최종의 다중경로 애니메이션에 대한 수정과 재작업에 많은 노력과 시간이 필요했을 것이다. 프로토타입의 캐릭터는 타이밍, 사양 및 상호작용의 관점에서 캐릭터들의 행동을 평가할 수 있는 비교적 단순한 도구였다. 뿐 아니라 애니메이션 제작자들은 고해상도 캐릭터 작업에 들어가기 전에 연습해볼 수 있었고, 피드백과 함께 완성할 수 있었다.(그림2 참조)

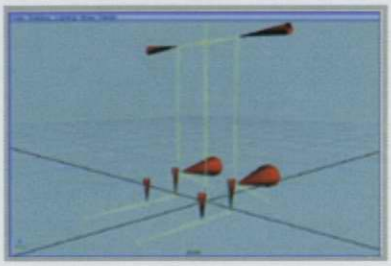
MEL 숏컷 : 셋업의 자동화

마야 임베디드 언어(Maya Embedded Language ; MEL) 스크립트는 원하는 복잡성 정도와 이를 구현하는 시간계획 사이의 격차를 해결할 수 있는 필수 요소였다. MEL 스크립트를 통해 셋업 작업의 단순화, 애니메이션 공정의 맞춤화 및 기술적 평준화가 가능했다.

이런 두 종류의 스크립트를 통해 구동키 기능의 장점을 최대한 활용할 수 있었다. 이것이 없었다면 애니메이션 제작이 어려웠을 것이며 일일이 수작업으로 하기에는 너무 지루했을 것이다. 또 다른 툴을 사용해 아티스트들은 기술적 경험에 관계없이 캐릭터들을 자동으로 IK 시스템에 맞출 수 있었다.

대부분의 두 발 캐릭터들의 다리 설정은 〈그림3〉과 같았다. 〈그림4〉에 나타나 있는 순서체계와 같이 다리는 표준형 엉덩이, 무릎, 발목관절 및 발뒤꿈치 관절, 2~3개의 발 뼈를 가지고 있다(여기서는 발의 뼈를 발가락으로 간주한다).

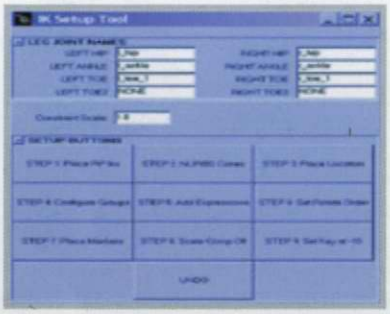
IK 장치는 3~4개의 RP(Rotate Plane) IK 핸들로 구성됐다. 이들은 엉덩이에서 발목, 발목에서 발가락, 발가락에서 발가락으로 연결돼 있거나 발가락이 아무 곳에도 연결되지 않았다. 이 모든 경우가 순서체계 〈그림5〉를 구성했으며 IK 핸들,



〈그림3〉 이러한 다리 설정이 대부분의 두 발 달린 캐릭터에 사용되어 각 캐릭터용 IK 시스템을 위한 성가신 수작업 설정을 피할 수 있었다.

〈그림4〉 하이퍼 그래프상에 표시된 캐릭터 다리의 표준 순서체계

〈그림5〉 하이퍼 그래프에 표시된 다리 조작 순서체계, IK 핸들, 로케이터 세트 및 몇 가지 NURBS 제약 대상 사이의 관계가 표시되어 있다.



〈그림6〉 IK 설정 툴(IK Setup Tool)은 반복적이고 오류가 쉽게 나타나는 설정 과정을 단순화 하였고 아티스트의 손에서 맞춤화되었다.

로케이터 세트 및 몇 가지 NURBS 제약 대상 사이의 관계를 나타내었다.

각각의 NPC, 적, 프로토타입에 대해 IK 시스템을 수동으로 설정한다면 많은 시간이 필요하다. 이런 시간은 이 캐릭터들을 최종적으로 만들어내는데 사용하는 것보다 낫다는 것을 알 수 있다. IK 셋업 툴을 사용해 긴 시간이 필요한 자질구레한 일은 단 몇 초에 간단히 마무리할 수 있다. 이 스크립트는 셋업 전문가를 필요로 하지 않았으며 비교적 단순한 코드는 아트 부서 내에서 완전히 맞춤화 할 수 있고, 정교하게 다듬을 수 있다.

IK 셋업 툴〈그림6〉은 3단계로 사용된다. 우선, 아티스트가 툴의 사전 설정 세트에서 캐릭터의 다리 관절 이름을 체크하고, 필요한 것을 수정한다. 다음에는 제약 대상용 치수가 캐릭터의 대략적인 크기에 근거해 입력된다. 그 다음에 9개의 버튼을 차례로 누르면 이 버튼들은 자동적으로 IK 핸들과 결합하게 되며 순간적으로 억제 위치체계를 구축한다.

IK 셋업 툴 분석

MEL은 변화가 많고 때로는 일관성이 없는 언어이다. IK 셋업 툴 개발에 걸린 시간의 상당한 부분은 문제에 대한 적절한 명령어를 찾는데 사용됐다. 9개의 툴 버튼이 담당하고 있는 핵심 과제를 작동하기 위해 필요한 MEL 명령어를 찾아냈다.

1번 버튼은 IK 핸들을 캐릭터의 다리에 위치시키는데 사용된다. 이 버튼은 조회모드(-q)에서 textFieldGrp 명령을 사용해 맨 위의 텍스트 필드로부터 뼈의 명칭을 찾아낸다. 이 스트링 변수들은 ikHandle 명령으로 전달되고 여기에서 IK 핸들을 만든다. 2번 버튼은 NURBS 콘을 캐릭터의 엉덩이, 발목 및 발가락 관절에 위치시킨다. 이들 콘은 MEL의 콘 명령어를 사용해 만들어지는데, 애니메이션 제작자들이 다리를 움직이는데 사용할 일차적 조작 대상이다. xform 명령은 다리뼈의 위치를 조회(-q)하고 이를 변수로 저장한다. 동작 명령은 이 변수를 읽고 콘을 움직여 각각의 위치에 배치한다. 마지막으로 MEL의 pointConstraint는 히프 콘을 캐릭터의 엉덩이에 결합시킨다.

CreateLocator라는 3번 버튼은 한 쌍의 로케이터를 각 장면에 배치한다. 다음에 그룹 명령어가 이 로케이터를 집단으로 묶고 xform(-q)이 캐릭터의 무릎 위치를 조회하고, 두 개의 새로운 모체를 무릎 관절로 이동시키며 로케이터들을 무릎 앞의 위치에 배치한다.

4번 버튼은 콘, 로케이터, IK 핸들, 모 그룹 및 제약을 부모 명령어를 사용해 표준화된 지위체계를 구성한다. 이 새로운 그룹은 move를 사용해 각 위치로 배치된다. 새로운 조작 관계가 무릎 로케이터와 주요 다리 IK 핸들, 조작 지위체계와 뼈대 사이에서 생성된다. 이들은 각각 poleVectorConstraint와 scaleConstraint 명령어를 사용해 구현된다.

5번 버튼은 각 장면에 몇 가지 표현을 추가해 데이터를 입력하는 수고를 덜어준다. 표현 명령어를 사용해 조작과 뼈대의 행동을 지정하는 표현 코드를 추가하면, 셋업 표현의 형성과 사양을 자동화할 수 있다. 6번 버튼은 setAttr을 사용해 발꿈치와 발가락 NURBS 콘을 XYZ로부터 YXZ로 회전 명령을 변경시킨다. 이 회전 명령은 색다른 Z-up 환경에서 가장 신뢰성 있는 회전을 생성한다.

7번에서 9번까지의 세 버튼은 최종 관리 기능을 수행한다. 7번 버튼은 polyCube와 부모 명령어를 사용해 커스텀 회전 가이드를 캐릭터의 척추로 모은다. 8번 버튼은 setAttr을 사용해 마야의 세그먼트 스케일 보상 스위치가 캐릭터의 모든 관절에 대해 차단되도록 한다. 마지막으로 9번 버튼은 setKeyframe을 사용해 -10에서 조회 프레임을 캐릭터

MEL 과정

```
// A method for querying a bone's position in world space:
xform -query -worldSpace -translation my_joint_name;

// A method for querying the contents of a text field:
textFieldGrp -query -text my_text_field_name;

// A method for setting a keyframe at frame -10 on a hierarchy:
setKeyframe -time -10 -hierarchy below my_hierarchy_name;

// Basic transformation methods: translation, rotation, and scaling:

// Moves an object to (0,0,5):
move -absolute 0 0 5 my_object_name;
// Rotates an object by 90 degrees on Z,
// relative to its current Rotation:
rotate -relative 0 0 90 my_object_name;

// Scales an object to 3 times its current size:
scale -relative 3 3 3 my_object_name;

// (Note: All flags are listed in their long forms.)
```

의 골격 및 제약 지위체계에 고정시킨다. 목록 1에 가장 유용한 것으로 판명된 몇 가지 MEL 과정을 제시했다.

MEL을 사용해 이 과정을 자동화함으로써 시간이 절약되고 수작업시 오류를 일으키기 쉬운 단계들을 제거했다. 또한 아티스트들이 셋업 경험에 관계없이 프로토타입과 캐릭터들을 IK 시스템에 신속하게 맞출 수 있게 돼 병목현상이 완화됐다. 이렇게 시간과 인적자원을 절약할 수 있게 돼 아트워크에 더 많이 집중할 수 있었다.

낮은 수준의 애니메이션 솔루션

지금까지 설명한 숏컷과 프로토타입은 좀더 나은 애니메이션을 효과적으로 생성하는 공동 목적을 가지고 있다. 이 방법들은 문제점들을 알아내고 시간을 절약함으로써 성취됐다. 그러나 때로는 첨단 기능의 솔루션을 사용하지 않고, 작업을 수행했다. 첨단 솔루션의 한정성, 비효율성 및 복잡성 때문이었다.

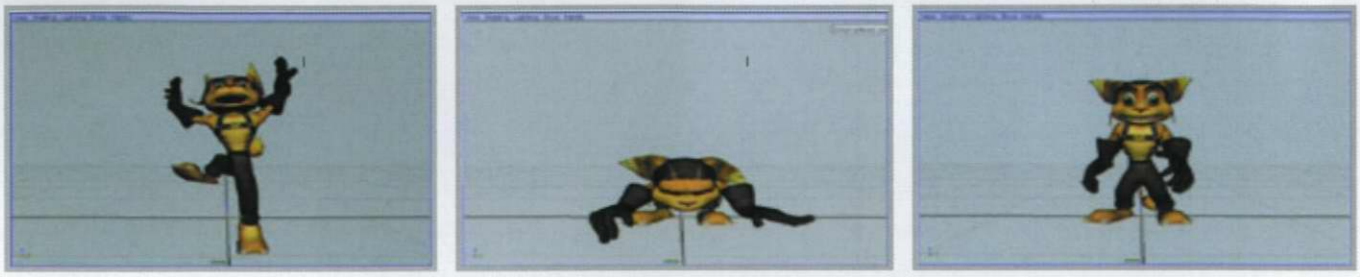
캐릭터들의 뼈대들은 지속적이고 반복적으로 해석되고, 크기가 조정됐다. 관절 부분 크기 조정과 이에 대한 해석을 통해 애니메이션에 느리고 빠른 움직임들이 추가됐다. 주요 해석은 2 프레임 정도 지속돼 디즈니의 아이디어처럼 보이는 것보다 느껴지는 것이 더 많도록 했다.

캐릭터의 척추와 팔에는 이렇다 할만한 IK 셋업을 갖추고 있지 않았기 때문에 이 부분의 뼈를 해석하는 것은 매우 단순했다. 필요하다면 이 관절을 조작하기 위해 다리의 IK 솔버 (solvers)를 차단할 수도 있었다. 해석과 크기조정은 전반적으로 효과적이었으며, 걷기, 공격 및 안면부 애니메이션 등 모든 면에서 결과가 좋았다. 〈그림7〉

추가의 셋업이 필요하지 않기 때문에 낮은 기술 수준의 솔루션을 사용해 시간을 절약할 수 있었다. 기능이 제한돼 있지만 이런 애니메이션 방법은 캐릭터의 자세를 만드는데 직접적이고, 입체감이 있으며 편리한 방법을 제공했다. 비록 외관이 보기에 좋지는 않지만 이 기법은 자원을 보존하고 애니메이션을 향상시키는데 있어서 다른 어떤 기법보다 효과적이었다.

걷기 및 보행 가이드

또 다른 애니메이션 보조도구로는 보행 가이드가 있다. 이 툴은 캐릭터가 걷고 뛸 때 발이 지면에 닿게 만드는데 사용됐



〈그림7〉 관절 치수 조정 및 해석을 통해 애니메이션 제작자들은 직접 자세를 조작할 수 있었고 저 비용의 기존기술을 사용하여 캐릭터를 더욱 생생하게 만들 수 있었다.

다. 비록 게임의 세계에서 발이 미끄러지는 것은 통상적으로 용납이 되고 있지만, 이런 현상을 제거함으로써 캐릭터의 움직임에 새로운 차원의 신뢰성을 줄 수 있었다.

보행 가이드는 입체모양에 부착돼 있는 작은 입체형상들과 함께 그 입체 모양을 길게 늘이는 효과를 준다. 작은 입체형상들은 캐릭터의 발목 및 발가락의 다각형 마커와 동일하며 셋업 과정에서 발에 모아진다.

특수한 모체 노드의 크기를 조정함으로써 이 가이드의 작은 입체형상들은 조정이 돼 캐릭터의 발 크기와 조화가 된다. 제약 및 로케이터 세트는 보폭이 변하더라도 이미 설정된 발의 크기를 일정하게 유지했다. 보행 사이클이 완성되고 나서는 캐릭터의 발 위치를 추적하면서 전방의 움직임을 시뮬레이션 하는 방법이 필요하다. 해결책은 디자이너가 지정한 속도에 보행 가이드를 맞추는 것이다.(예를 들면 초당 2미터로 지정할 수 있다) 일단 보행 가이드가 적절한 속도로 움직이게 되면 소형 입체형상들은 크기가 정확히 조정되고, 애니메이션 제작자들은 캐릭터의 걸음 사이클 작업을 시작할 수 있다.

보행 가이드를 사용해 발이 미끄러지는 것을 제거하기 위해 사용된 기법은 캐릭터의 발 마커를 가이드의 소형 입체형상과 나란히 정렬하는 것이었다. 이 방식은 발이 지면에 접촉하는 모든 프레임에 적용됐다. 〈그림8〉

사이클이 완료되자 캐릭터는 한 레벨에 적용돼 발의 미끄러짐이 거의 없이 이미 설정된 속도로 움직였다. 또한 프로그래머는 발의 지면 접촉을 유지하면서 캐릭터의 속도에 따라 사이클의 재생 속도를 조정할 수 있다. 몇 가지 게임 플레이 상황은 테스트 케이스처럼 명확하지 않다. 그러나 보행 가이드는 캐릭터의 발을 적절한 위치에 놓아주었다. 일단 이 가이드에 익숙해지면서 운동의 기술적 측면을 더 많이 추적할 수 있게 돼 애니메이션 제작자들은 작업 일정뿐 아니라 미술작업에도 큰 도움을 받았다.

안면부 만들기 : 미술적 이유 및 기술적 세부사항

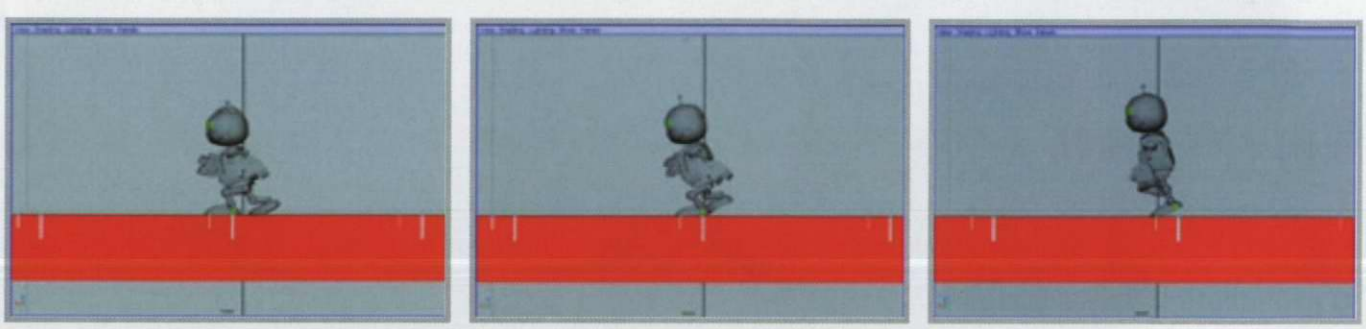
Ratchet & Clank 개발 초기부터 표정이 화면 뿐 아니라 게임플레이 애니메이션에도 중요한 요소라는 것을 알고 있었다. 표정의 문제에서도 애니메이션의 심도와 일정상의 효율성이라는 문제를 동시에 해결해야 했다. 안면부를 만드는데 두 가지 방법을 결정했는데 적은 단순하게 했고, 주인공은 좀더 복잡하게 했다. 표정은 한가한 동작을 과장했고 공격은 강화했다. 적에 대한 애니메이션 작업 시에는 기존의 애니메이션 방식을 이용했다. 통상적으로 애니메이션을 보는 사람은 캐릭터의 얼굴, 특히 눈을 주시한다. 캐릭터의 눈과 입에 대한 주의 집중은 특히 빠른 게임플레이 사이클 중에 확실한 동작을 만들어 내는데 매우 중요하다.

대부분의 적 캐릭터는 상당히 단순한 얼굴 골격을 가지고 있다. 그러나 이런 골격들은 고도로 조작 가능한 눈과 입을 가지고 있다. 각 눈은 2~4개의 뼈대를 가지고 있으며, 눈썹과 눈꺼풀 조작이 가능하다. 입은 일반적으로 조금 더 단순해 1개 또는 2개의 뼈대를 가지고 있다. 대부분의 경우 이런 설정을 통해 융통성 있게 적의 모습을 과장하고 그 행동의 감정을 고조시킬 수 있었다. 〈그림9〉

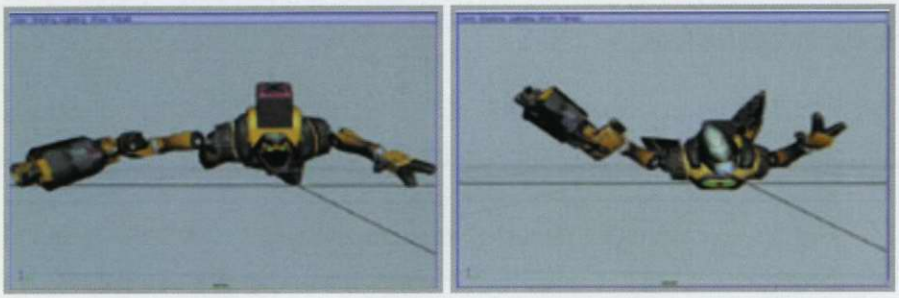
주인공들의 안면부 설정은 좀더 복잡해 NPC를 공유했다. NPC의 얼굴이 대부분 영상기술에서 조작됐지만 Ratchet & Clank 또한 게임 플레이 중에 표정을 많이 사용했다.

적에 대한 설정과 마찬가지로 주인공과 NPC 얼굴은 안면부 관절을 통해 조작됐다. 이 관절들은 적의 경우와는 달리 직접 변형된 것이 아니라 구동키 시스템을 통해 애니메이션으로 제작됐다. 이들이 스크린에 등장하는 시간이 좀 더 많기 때문에 주인공 및 NPC 얼굴은 적의 경우보다 훨씬 더 많은 뼈와 표정을 갖게 되는 경향이 있었다.

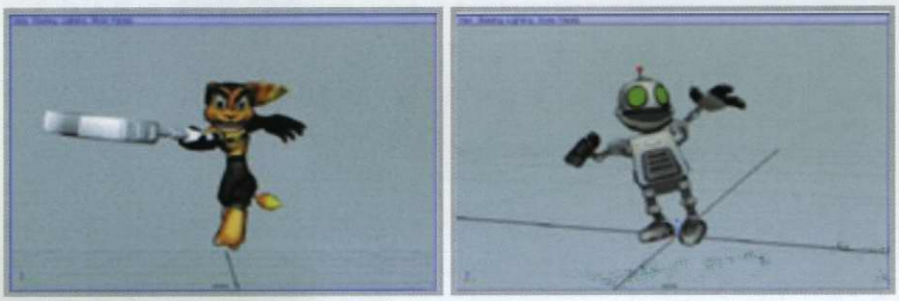
〈그림10〉을 보면 게임 플레이 중에 Ratchet and Clank가 만들어 내는 표정의 범주를 알 수 있다. 마음이 들뜨면 웃고, 공격을 당하면 찡그리며, 싸움을 하는 중에는 이를 악물고, 추



〈그림8〉 보행 가이드(Walk Guide)는 프레임에서 캐릭터의 발을 지상에 적절히 배열함으로써 발이 어색하게 미끄러지는 현상을 최소화 하였다.



〈그림9〉 적의 안면부 골격에는 적은 것으로 많은 효과를 얻었다. 뼈의 세부사항을 줄여서 눈과 입에 사용함으로써 단순하면서도 과장된 표정이 가능하였다. 게임상의 애니메이션에서 로봇 낙하산 부대원이 맞아서 넘어진 모습이 얼굴에 표현되어 있다.



〈그림10〉 Ratchet와 Clank의 게임 플레이 안면 애니메이션 시스템은 게임 플레이 동안에 필요한 표정이 광범위하기 때문에 적들을 표현하는 것보다 정교한 설정 작업이 필요하다.

우면 이가 딱딱 부딪치며 죽으면 턱이 축 늘어진다. Clank의 표정은 Ratchet의 등에 업혀 있을 때와 따로 플레이 할 때로 바뀐다.

앞에서 언급한 바와 같이 주인공과 NPC의 얼굴 표정은 MEL 스크립트 슬라이더 인터페이스를 통해 이미 설정된 구동키 속성들을 조합해 구성됐다. 이런 사전 설정을 이용하면 애니메이션 제작자들은 아무것도 없는 상태에서부터 얼굴표정을 구축할 필요 없이 다양한 표정을 조합하거나 만들어 낼 수 있다. 마치 색상표처럼 속성들을 함께 섞어서 새로운 조합을 만들어 낼 수 있는 것이다.

40여 종류의 캐릭터 안면부 속성 중에 절반 정도는 얼굴 전체 또는 일부의 기본 표정을 만드는데 사용된다. 이러한 기본 표정에는 분노, 혐오와 공포, 행복, 슬픔, 놀람 등이 포함돼 있으며 이 모든 것은 플레이어가 쉽게 인식할 수 있다. 좀더 미묘한 속성들은 발음을 표현하고 각각의 얼굴 모습을 통제하는데 사용됐다. 독특하고 다양한 감정의 범위는 표정, 발음 및 외형 속성물을 함께 조합해 만들어낼 수 있었다.

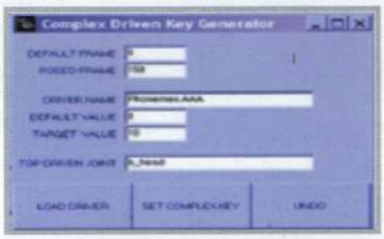
안면부 사전설정 스크립트 작업

캐릭터에 안면부를 사전 설정하는 작업에는 어느 정도의 설정 시간이 소요됐다. 그러나 다른 MEL 스크립트를 사용해 이 과정의 일부를 최적화할 수 있었다. 다른 MEL 툴처럼 이 스크립트는 몇 가지 귀찮은 과정을 자동화해 셋업 아티스트들이 얼굴 자세를 만드는데 더 많은 시간을 할애할 수 있었다.

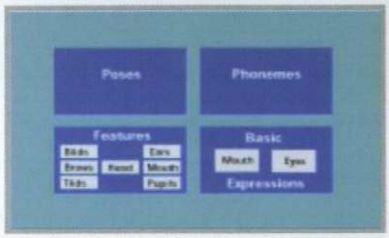
안면부 사전설정은 별도의 애니메이션 파일로 만들어지며, 여기서 각각의 표정, 발음 및 외관의 자세 등은 별도의 키 프레임으로 저장된다. 이 파일이 완성되면서 캐릭터 아티스트는 MEL 구동 키 생성기〈그림11〉를 사용해 각 자세에 대한 구동키를 자동으로 설정했다.

구동 키 생성기는 키 프레임에 의한 자세의 변형과 디폴트 자세를 비교한다. 채널이 디폴트로부터 변형된 것을 스크립트가 등록하면 변경 값에 기초해 채널의 구동키를 설정한다. 이 스크립트는 MEL의 산술적 기능에 의존해 변경값을 확인하며, setAttr 및 setDriven Keyframe 명령어가 드라이버를 활성화한다. 목록 2에 몇 가지 구동키 생성기의 단순한 코드를 제시했다.

안면부 애니메이션의 드라이버는 〈그림12〉에 나타나 있는 컨트롤 박스라는 모델에 저장됐다. 이 입방체의 순서체계는 안면부 속성의 가시적 외형 역할을 하며, 제2의 인터페이스로서 2배가 될 수 있다. 작업의 효율성을 높이기 위해 Ratchet과 Clank, 모든 NPC 캐릭터들은 동일한 컨트롤 박스를 가지고



〈그림11〉 구동 키 생성기는 사전에 설정된 얼굴의 포즈를 분석하여 이를 중립 포즈와 비교하고, 구동 키를 변화된 채널에 할당한다.



〈그림12〉 각 NPC 및 주인공은 각각의 컨트롤 박스를 갖추고 있었으며, 이곳에 안면부 드라이버가 저장되었다. 안면부 드라이버는 실질적으로 컨트롤 박스 입방체의 속성이다.

구동 키 생성기의 샘플 코드

```
// The "if" gate checks for changed X-Translation values
// between the Default and Posed frames.

if ($txa != $txb)
{
        // Sets the Driver Attribute and the Current Joint's
        // X-Translation to their Default Values;
        // Sets a Driven Key Frame for the Default Values.

        setAttr $atnm $dr0;
        setAttr ($current + ".tx") $txa;
        setDrivenKeyframe -currentDriver $atnm -attribute
        translateX $current;

        // Sets the Driver Attribute and the Current Joint's
        // X-Translation to their Posed Values;
        // Sets a Driven Key Frame for the Posed Values.

        setAttr $atnm $dr1;
        setAttr ($current + ".tx") $txb;
        setDrivenKeyframe - currentDriver $atnm -attribute
        translateX $current;

        // Prints command summary to the Script Editor for
        // easy reference.

        print ($current + ": TX has been keyed for slider
        value 0: " + $txa + " and slider value 10: " +
        $txb); print "\n";
}

// In this loop segment, $current is the current joint,
// and $atmn is the attribute name. $dr0 and $dr1 represent
// Default and Posed Driver values. $txa & $txb are the
// Default and Posed X-translation values, respectively.

// (Note: All flags are listed in their long forms.)
```

있었으며 Ratchet의 경우에는 훨씬 더 많은 능동형 드라이버를 가지고 있었다.

자동화 설정 방법은 세 가지 이유 때문에 애니메이션 작업에 이점을 제공한다. 첫째, 이 방법을 사용할 경우 아티스트들은 수작업으로 주요 뼈대, 채널, 드라이버를 확인할 필요가 없다. 둘째, 이 방식은 구동키를 변경된 채널에만 할당하며 영향을 받지 않은 채널들은 키 프레임에 대해 애니메이션 제작자들이 손 댈 필요가 없다. 마지막으로 이 방식은 마야의 내장 구동키 인터페이스에 선행한다. 이제까지 여러 뼈와 채널을 동시에 한 드라이버에 할당하는 것이 부담스러웠을 뿐 아니라 신뢰성도 낮았다.

방법에 관계없이 안면부 애니메이션 제작은 게임 플레이 캐릭터들에 있어서 매우 중요한 역할을 했다. MEL은 아티스트들이 발전된 마야의 사양에 접근하고 작업의 흐름을 최적화하는 도구가 됐다. 주인공이든 적이든, 이 게임에서 실제로 모든 캐릭터의 성격은 얼굴 표정을 통해서 강화됐다. 또한 플레이어뿐 아니라 캐릭터 사이의 상호작용이 향상됐다.

사이클의 종료

모든 캐릭터 구동 프로젝트와 마찬가지로 Ratchet & Clank는 애니메이션 작업팀에게 독특한 아트 및 기술적 문제점들을 제시해 주었다.

애니메이션 작업은 이 캐릭터들을 통해 플레이어들이 제작자들의 세계를 경험한다는 이해에 기초를 두고 있었다. 이런 문제점들을 해결함으로써 이 애니메이션의 캐릭터들은 단순한 게임 공간에서 벗어나 플레이어들이 동일시하고 헐뜯고 인격화하는 실체가 된다.

프로젝트를 완성하기 위해서 이런 철학은 실용적인 방법론과 결합돼야 한다. 이런 필요성 때문에 테스트 방법, MEL 숏컷 및 실시간 애니메이션 공정 등을 개발했다. 제작에 있어서 이런 방법들은 많은 장애 요인들을 제거했다. Ratchet & Clank 작업 중에 개발한 시스템과 공정들은 지속적으로 조정 및 확장될 예정이다.