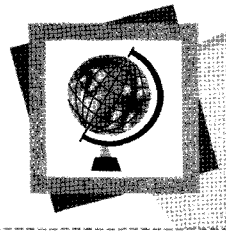
| 특집 06 |



# 게임 그래픽

김일혁 손홍정· 정택상
((주)오토데스크 코리아) (고려대학교)

목 차



## 1. 서 론

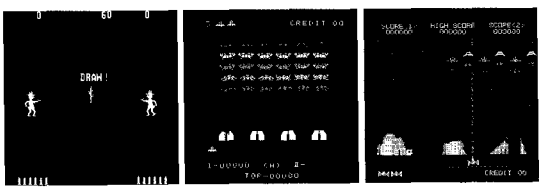
게임은 21세기 첨단 산업을 대표하는 분야 중의 하나로서 디자인, 기획, 프로그래밍, 그래픽 등 다양한 요소들의 유기적 관계를 통해 제작된다. 이 중 게임의 얼굴이라고도 불리는 그래픽은 게임과 유저를 이어주는 1차적인 경로가 되며, 유저들의 호감 및 흥미를 이끌어 내는데 큰 역할을 한다.

현대와 같은 초스피드 사회를 살아가는 사람들은 1차적인 시각적 효과를 중시하고, 어느 정도의 자극을 받지 않으면 반응을 보이지 않는 수동적인 특성이 있다. 게임 유저들은 훌륭한 게임성, 탄탄한 스토리 등의 게임 요소 보다는 먼저 화면을 통해 자극을 받고 느끼며 반응한다. 이러한 특성으로 인해 유저들은 더 나은 그래픽 퀄리티를 요구하게 되고, 게임 제작 업체들은 그러한 요구에 부응하기 위해 고해상도 모드 게임 제작, 퀄리티 향상을 위한 투자 확대 등을 통해 더욱 부드럽고 화려하며 사실적인 게임 그래픽을 표현하기 위해 노력하고 있다.

## 2. 게임 그래픽의 역사

게임 그래픽은 게임의 역사와 함께 해 왔다고 해도 무관하다. 1962년 Space War라는 미국 게임이 컴퓨터 게임의 시초였으며, 그 당시 그래픽은 (그림 1) (a)에 보여진 바와 같이 아주 단순한 8bit 표현 방식의 그래픽으로 출발했다.

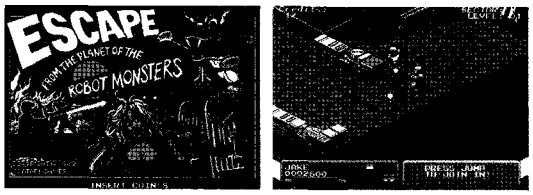
70년대 중반을 거치면서 단색 톤의 게임이 아닌 컬러 게임들이 등장하게 된다. (그림 1) (b)와 (c)는 이 시기에 출시된 대표적인 게임인 갤럭시안과 갤러그의 스크린 샷이다.




(a) (b) (c)

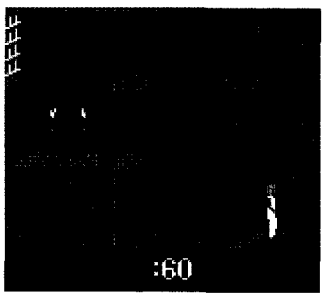
(그림 1) 6,70년대 초창기 게임

80년대에 접어들면서 수많은 오락실용 게임 즉, 아케이드 게임들이 다수 등장하게 된다. 80년대 초반에 국내 최초의 게임인 "신검의 전설 라이어"이 등장하게 된다(그림 2 참조).



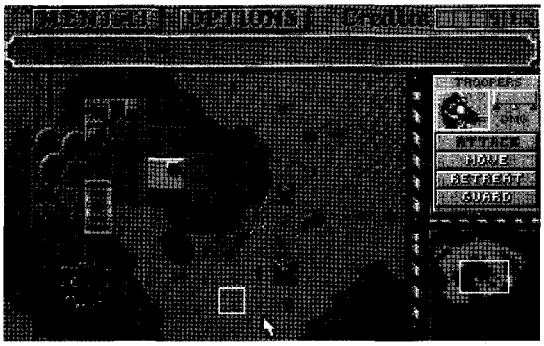

(그림 2) 80년대 초창기 게임

이후 16bit 기반의 개인용 컴퓨터 시대가 열리게 되었고, 이에 따라 월등히 향상된 그래픽과 음향 효과 등을 도입한 게임들이 속속 출시되었다. 특히 IBM이 출시한 "페르시아의 왕자"(그림 3)는 이전 게임들보다 부드러운 동작의 애니메이션 그래픽과 다양한 음향 효과로 게이머들을 매료시켰다.




(그림 3) 페르시아의 왕자

90년대에는 비디오카드와 사운드카드의 비약적인 발전과 더불어 게임의 중흥을 보게 된다. 이 시절에는 "원숭이 섬의 비밀", "인디아나 존스" 등의 어드벤처 게임의 대약진이 이루어졌던 시대이다. 그 후 "심시티", "듄2"와 같은 전략시뮬레이션 게임이 등장하면서 새로운 방식의 그래픽이 표현되기 시작하게 된다. 특히 (그림 4)의 "듄 2"는 실시간 전략 게임의 시초로서의 의미 역시 가지고 있다.

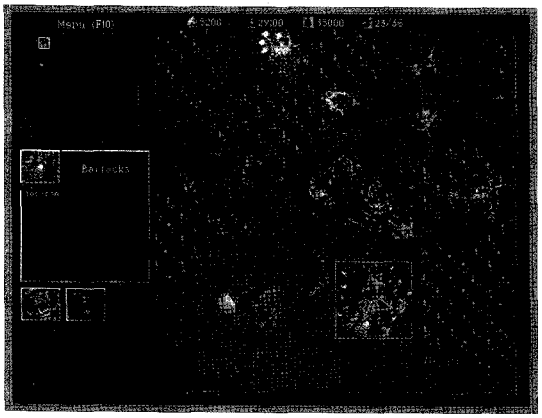



(그림 4) Dune 2

그러나, 시뮬레이션 게임의 약진보다는 "울펜스타인", "둠" 등의 1인칭 시점 게임(FPS)의 성공이 게임 그래픽 분야에 커다란 영향을 미쳤다. (그림 5)는 "둠"의 스크린 샷인데, 지금 보기에는 조악해 보일 수도 있으나 당시 사용자들이 받은 시각적 충격은 대단한 것이었다. 이 시기를 기점으로 게임 그래픽에 3차원적인 요소가 대거 추가 되었으며 90년대 중반부터 좀 더 퀄리티 있는 그래픽을 위해 3차원 그래픽 소프트웨어를 이용한 개발이 본격적으로 시작되었다.

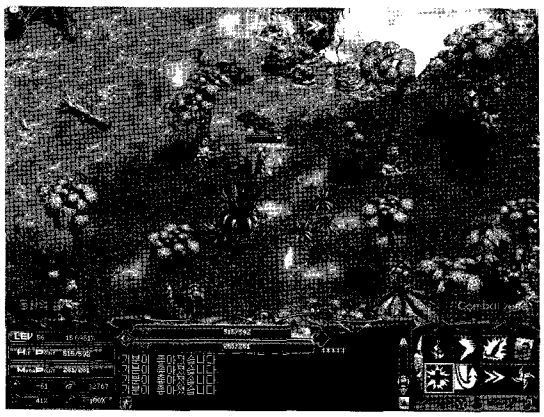

(그림 5) Doom

하나의 예로, 90년대 중반에 나온 "Command and Conquer", "WarCraft" 시리즈 등은 2D게임이지만 3D렌더링을 통해 얻은 이미지를 사용하는 등 게임 그래픽의 수준을 한차원 높여 놓았다. (그림 6)은 이 시기 출시된 대표적 게임 WarCraft II이다.



(그림 6) WarCraft II

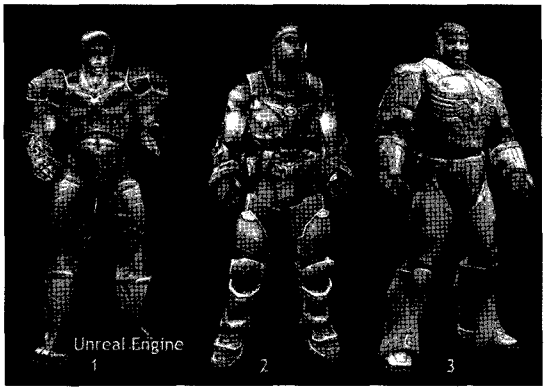
90년대 후반 인터넷 시대에 접어들면서 2차원 롤 플레잉(role playing) 온라인 게임들이 쏟아져 나오기 시작했다. 게임 그래픽 역시 그에 비례해서 비약적인 발전을 하게 된다. 이 시기 대표적인 온라인 게임은 "울티마 온라인"이 있고, 그 이후 국내에서 아직까지도 선풍적인 인기를 끌고 있는 "리니지"(그림 7)가 있다.



(그림 7) 리니지

2000년대로 접어들면서 게임에 대한 장르가 파괴되면서, 게임의 특성을 보다 선명히 드러내고 사용자들에게 어필하는 요소로서 그래픽적인 요소가 더욱 부각된다. 이 과정을 통해서 그래픽 분야에서도 3차원이 주를 이루게 되면서 게임유저들은 더욱 정교한 이미지와 부드러운 애니메이션을 즐길 수 있게 되었다.

지금까지 아케이드와 PC게임 위주로 그래픽 발전사에 대해 알아보았다. Playstation이나 Xbox용 콘솔 게임의 경우 PC 게임보다 우수한 퀄리티의 그래픽이 표현되는 것을 볼 수 있다. (그림 8)에서 보는 바와 같이 콘솔게임의 캐릭터 또한 발전되어 왔다.




(그림 8) Unreal Character

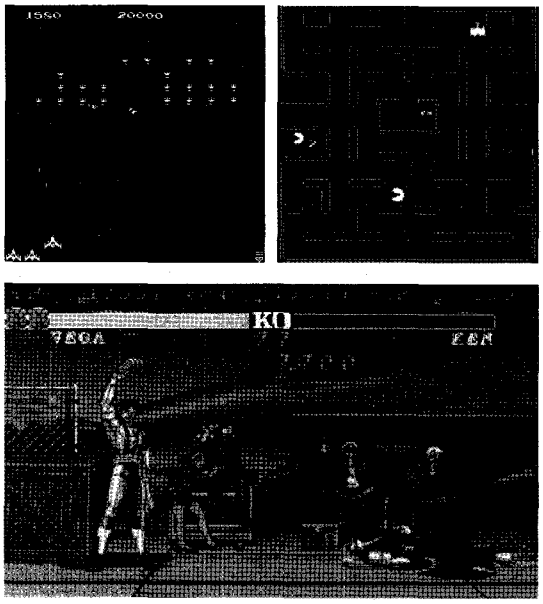
앞으로의 게임 그래픽은 하드웨어의 발전과 더불어 좀 더 사실적이 될 것이고, 이러한 사실적인 디테일을 표현하기 위해 또한 게이머들의 요구를 충족시키기 위해 좀 더 많은 표현 방식들이 사용될 것이다.

## 3. 2D 그래픽과 3D 그래픽의 차이

지금까지 게임 그래픽이 역사적으로 어떻게 발전되어 왔는지 살펴보았다. 그렇다면 2D 그래픽과 3D 그래픽은 어떻게 다른지 알아보기로 하자. 2D 게임의 그래픽 데이터를 얻어내기 위해서는 두 가지 방식이 사용된다. 첫 번째는 2D 그래픽 소프트웨어만을 이용하여 도트(dot) 작업 방식을 통해 이미지를 얻는다. 이러한 그래픽 데이터를 이용해 제작된 게임으로는 갤러그, 팩맨, 스트리트 파이터 등이 있다. 두 번째 방식에서는 3D 소프트웨어의 렌더링 과정을 통해

이미지를 얻는다. 하드웨어의 발전과 더불어 게임 상에서 표현할 수 있는 색상의 수도 늘어났는데, 16컬러만 지원되던 단계에서 256컬러가 지원되면서부터 3D 소프트웨어를 이용한 2D 이미지 제작이 시작되었다.

(그림 9)부터 (그림 11)까지는 각각 게임의 종류를2D 게임, 3D 소프트웨어로 생성된 이미지를 이용한 2D게임, 실시간 3D 게임으로 나누어 비교해 본 것이다.




(그림 9) 2D 게임 (갤러그, 팩맨, 스트리트 파이터)

(그림 10) 3D 소프트웨어를 이용한 2D 게임 (스타크래프트)

(그림 11) 실시간 3D 게임 (리니지 II)

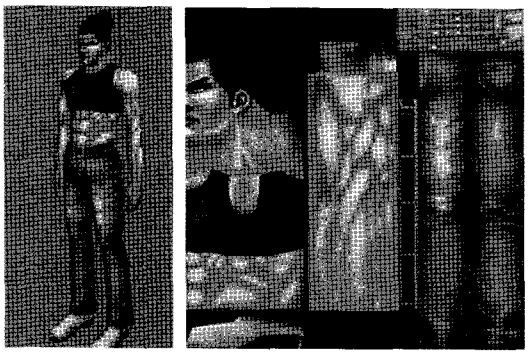
위 그림들을 통해 알 수 있듯이 2D 게임은 정해진 2D 이미지를 스프라이트(sprite)의 형태로 화면에 표현한다. 3D 소프트웨어를 이용해 만들어진 2D 이미지를 이용하는 경우는 이미지(스프라이트) 소스를 빠르게 만들 수 있음은 물론 다양한 각도에서의 이미지를 쉽게 얻어낼 수 있기 때문에 많이 사용되었다. 그러나, 이 방식은 화면 상의 뷰(시점)에 대한 조작이나 캐릭터들의 움직임 등을 표현하는 데 있어서는 한계가 있다. 따라서, 실시간 3D 엔진을 이용한 3D 게임이 각광을 받기 시작했으며, 최근 소개되는 게임들은 이러한 실시간 3D 엔진을 이용한 것들이 대부분이다.

최근 PC를 비롯한 게임 하드웨어 플랫폼 성능의 비약적 발전에 기반하여, 고품질 그래픽 기법들이 널리 쓰이고 있다. 다음 절에서는 대표적인 고품질 그래픽 기술에 대해 살펴보자.

## 4. 현재의 게임 그래픽 기술

최근에 만들어 지고 있는 실시간 3D 게임들에는 여러 가지 기술적인 표현이 많이 추가되어 있다. 실시간 그림자에서부터, 반사 표현, 부드러운 애니메이션, 하이 폴리곤 (high polygon) 캐릭터, 라이트맵 (light map), 노멀맵 (normal

map), 여러 가지 특수효과를 더해주는 FX 쉐이더 등과 같은 여러 가지 표현 기법들을 이용해 게임 그래픽의 퀄리티를 한층 높이고 있다. 사용자들은 하드웨어 사양에 따라 여러 가지 설정을 선택적으로 적용시킬 수 있다. 예를 들어, 그림자의 경우, 단순한 원형 형태의 그림자와 실시간 그림자를 선택할 수 있으며, 텍스처 또한 사이즈가 작은 것부터 큰 텍스처까지 선택이 가능하다.



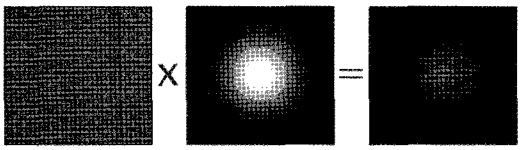
(그림 12) 텍스처가 적용된 캐릭터

그래픽 퀄리티 향상을 가져온 핵심적 요소 기술 중에는 텍스처를 이용한 매핑 기법이 있다. 텍스처 매핑을 위해서는 메쉬의 각 버텍스마다 적절한 텍스처 좌표를 가지고 있어야 한다. 2차원 텍스처를 이용하는 경우, 이를 통상 (u, v) 좌표라 부른다. 이러한 텍스처 좌표를 이용하여 텍스처의 특정 위치에 접근해 RGB 색상 등과 같은 텍스처 정보를 추출한다. (그림 12)는 텍스처가 입혀진 캐릭터와 2차원 이미지 텍스처를 보여준다.

이러한 매핑 기법을 한 단계 진척시킨 기술로, 그간 꾸준히 사용되어 온 라이트맵과 최근 사용이 급증한 노멀맵이 있는데 이에 대해 조금 더 구체적으로 살펴보겠다.

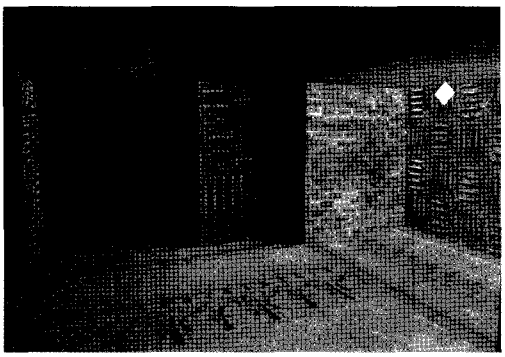
### 4.1 라이트 매핑

라이팅은 게임에서 매우 중요한 부분이다. 라이팅 모델은 여러 가지가 있는데, 광원의 방향과 물체의 특정 위치에서의 법선 벡터 등을 이용한 비교적 간단한 모델에서부터 광선추적법(ray-tracing)을 통해 빛의 반사까지 고려한 복잡한 모델까지 다양하다. 본 학회지의 "3차원 렌더링" 원고에서 이미 다룬 대로 일반적인 라이팅은 버텍스별, 또는 픽셀별로 연산할 수 있는데, 라이팅 연산의 양을 줄이기 위해 라이트맵이라는 기법이 널리 쓰이고 있다. 라이트 맵은 라이팅이 적용된 결과를 텍스처로 저장해 놓은 것인데, 실제 렌더링 단계에서는 라이팅 연산을 하지 않고 단순히 텍스처에서 값을 읽어오는 연산으로 라이팅을 대체할 수 있다는 장점이 있다. 따라서 실




(그림 13) 라이트맵

(a)



(b)

(그림 14) 라이트맵 적용 전과 후

시간 렌더링에 매우 유용하다. 단, 이는 광원과 물체가 모두 움직이지 않는 경우로 제한된다.

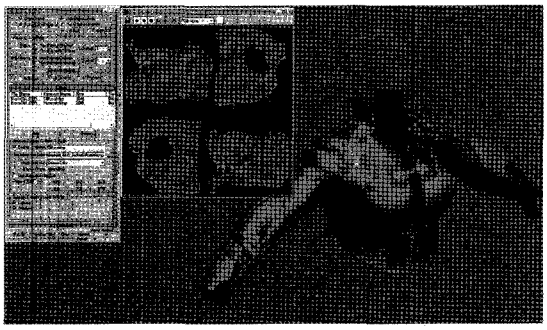
(그림 13)에서 왼쪽 그림은 물체 자체의 매터리얼을 통해서 얻어진 색이고, 가운데는 고정된 광원에 대해서 비추어진 조명 정도를 저장한 라이트맵이다. 이 두 정보를 합하여 오른쪽 (그림 14)와 같은 최종 결과를 만들어 낼 수 있다. 그림 14) (a)는 물체의 매터리얼만을 통해서 얻은 영상이고, (그림 14) (b)는 라이트맵을 추가로 적용한 것이다.

## 4.2 노멀 매핑

노멀맵은 로우 폴리곤 (low polygon) 캐릭터나 배경을 사용하되 마치 하이 폴리곤 캐릭터나 배경을 렌더링한 것처럼 표현해 주는 방식이다 (본 학회지의 "3차원 렌더링" 원고를 참고하라). (그림 15)는 로우 폴리곤 캐릭터에 노멀맵을 적용해 렌더링한 결과이다. 그림에서 볼 수 있듯이 로우 폴리곤 모델로도 높은 퀄리티를 얻을 수 있다.

(그림 15) 노멀맵을 이용한 로우 폴리곤 렌더링

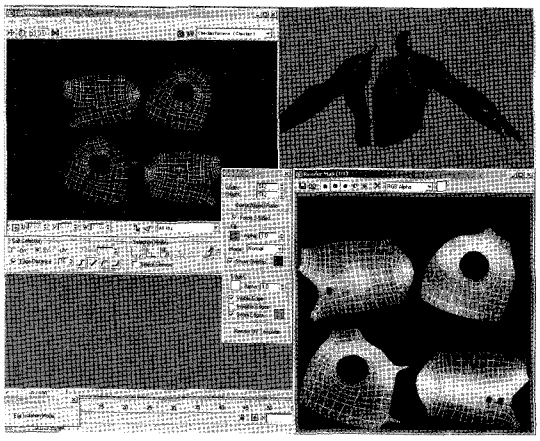
노멀맵이란 폴리곤 메쉬 표면의 울퉁불퉁한 정도를 나타낸 것이다. 표면이 울퉁불퉁함에 따라 그 표면의 법선 벡터 값 역시 달라지게 되는데, 이 법선 벡터를 비트맵으로 저장한 것이 노멀맵이다. 보통 울퉁불퉁한 디테일은 하이폴리곤을 써야 표현 가능하지만, 로우 폴리곤을 사용하고도 같은 효과를 가질 수 있도록 하이 폴리곤을 통해서 노멀맵을 얻어 둔 다음, 로우 폴리곤을 렌더링하면서 노멀맵의 법선벡터를 적용해주면 로우 폴리곤으로도 울퉁불퉁한 디테일을 살릴 수 있는 것이다.



(그림 16) 노멀매핑을 위한 3ds Max 작업

최근 3D 소프트웨어의 진보와 더불어 이러한 노멀맵 데이터를 쉽고 간단하게 만들어 낼 수 있다. (그림 16)은 3ds Max를 통해서 노멀맵을 만들어 내는 과정의 한 장면을 보여주는데, 노멀맵은 두 단계를 거쳐 생성된다. 첫번째는 로우 폴리곤 모델과 하이 폴리곤 모델을 매칭시켜주는 과정이다. 3ds Max 8에서는 이 매칭과정을 위해서 케이지(Cage)라는 도구를 제공하고 있으며, 이것을 이용하면 간단한 세팅만으로 매칭과정을 자동으로 수행할 수 있다. (그림 16)의 오른쪽 부분은 이 케이지 도구를 이용하는 모습이다.

두 번째 과정은 하이 폴리곤 모델의 노멀벡터들을 대응된 로우 폴리곤 모델의 버텍스에 텍스처 좌표 (u,v)로 기록해주는 것이다. 접선벡터는 (x,y,z) 성분을 갖는 방향벡터이고, 이것을 텍스처의 (r,g,b) 성분에 변환 저장한다. (그림 17)은 3ds Max로 이 과정을 수행한 결과로서, 오른쪽 하단이 생성된 노멀맵이다.



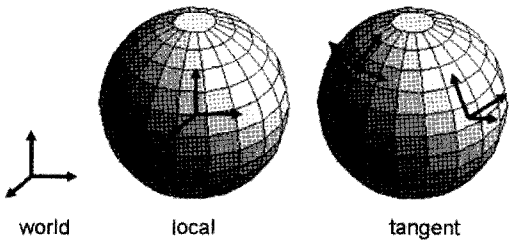
(그림 17) 3ds Max에서 생성한 노멀맵

노멀맵은 노멀이 정의된 기준 좌표계에 따라서 다음과 같은 4가지 형태로 생성이 가능하다. World 좌표계를 기준으로 한 노멀맵은 말 그대로 전체 scene을 표현하는 world 좌표계에서 정의된 것이며, screen 좌표계 노멀맵은 게임 내 카메라 좌표계에서 정의된 것을 말하며, local 좌표계 노멀맵은 각 object가 가지고 있는 local 좌표계를 기준으로 접선 벡터를 표현한 것이다.

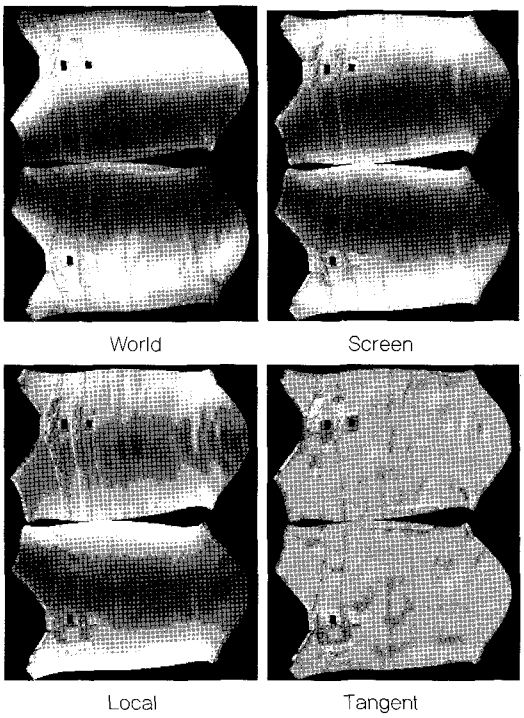
한편, tangent 좌표계 노멀맵은 object 표면의 각각의 지점에서의 접선 공간(tangent space)을 기준으로 접선 벡터를 표현한 것이다. 접선 공간이란 삼각형의 접선(tangent), 종법선(binormal), 법선(normal) 벡터가 x, y, z 축으로 표현되는 좌표계를 말하는 것이다(그림 18 참조).

〈표 1〉 노멀맵 종류

| | |
|---|---|
| World | Transform이 적용되지 않는 오브젝트<br>예) 건물, 가구 등 |
| Screen | Billboard 개념을 쓰는 오브젝트<br>예) 나무, 파티클 등 |
| Local | 강체변환만 적용되는 오브젝트<br>예) 자동차 |
| Tangent | 비강체 변환이 적용되는 오브젝트<br>예) 캐릭터 |



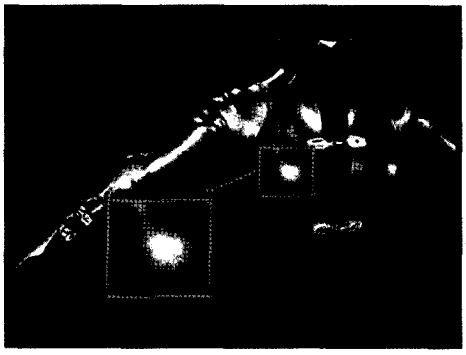

(그림 18) 월드, 로컬, 접선 좌표계




(그림 19) 노멀맵 가시화

(그림 18)은 네 종류의 노멀맵이 각각 어떠한 성격의 오브젝트에 적당한가를 보여준다. (그림 19)는 네 종류의 노멀맵을 가시화한 결과이다. 이중 tangent 좌표계 노멀맵을 보면, 다른 노멀맵과 달리 전체적으로 유사한 색조(파란색)를 가짐을 알 수 있다. 그 이유를 살펴보자. Tangent 좌표계에서 정의된 노멀은 대개의 경우 그 tangent 좌표계의 z축에서 약간씩 벗어나(deviated 또는 perturbed) 있는 것일 뿐, x나 y축에 더 가까운 것은 아닐 것이다. 즉, 대부분의 노멀은 (0,0,1)에 가깝게 된다. 이렇게 (0,0,1)에 가까운 노멀

을 (r,g,b)로 해석해서 가시화하면 b 즉, blue(파란색) 요소가 전체를 지배하게 되므로, (그림 19)와 같은 결과가 나오는 것이다. (그림 20)은 노멀 매핑을 이용하여 렌더링한 자켓의 모습이다.

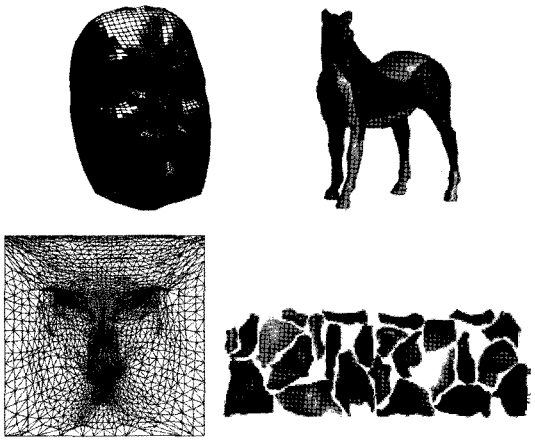


(그림 20) 로우 폴리곤 모델에 노멀맵이 적용된 결과

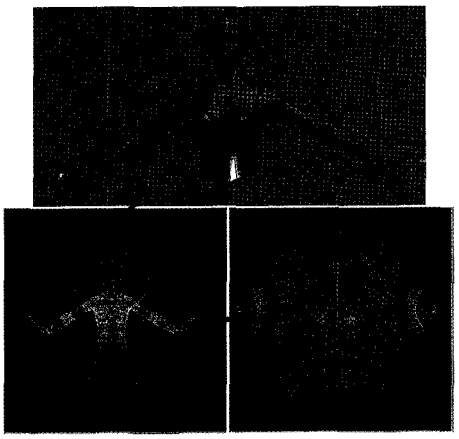
### 4.3 펠트 매핑

지금까지의 텍스처 매핑 기술은 메쉬의 각 버텍스마다 텍스처 좌표가 세팅되어 있다는 것을 전제로 한다. 그러나, 텍스처 좌표 값을 적절히 설정해 주는 것은 용이한 문제가 아니다. 왜냐하면, 메쉬는 3차원이고 텍스처는 2차원이기 때문이다. 이를 직관적으로 이해하면, 3차원 메쉬를 (그림 21)과 같이 2차원 평면에 펼치는 것과 동일한 문제가 된다. 이러한 과정을 파라미터라이제이션(parameterization)이라고 부르며, 3ds Max의 용어로는 UV warping이라고도 한다. 그 중에서도 (그림 21)의 오른쪽 하단부처럼 여러 개로 나누어진 메쉬가 하나의 텍스처에 파라미터라이제이션된 것을 텍스처 아틀라스(texture atlas)라 한다.

파라미터라이제이션의 중요한 기준은, 어떤 영역의 폴리곤들의 크기와 그에 매핑되는 텍스처 크기의 비율이 다른 영역에서의 그것과 최대한 유사하게 되어야 한다는 것이다. 이것이 보장되지 않으면, 매핑된 텍스처가 일부 메쉬 영역에서는 너무 확대되고 다른 영역에서는 축소되어 전체적으로 부자연스러운 결과를 내기 때문이다. 적절하게 파라미터라이제이션을 수행하기 위해서 다양한 수치적 방법들이 사용되는데, least square를 이용하는 방법[1]과 pelt mapping [2]을 이용하는 방식 등이 있다.



(그림 21) 파라미터라이제이션



(그림 22) Pelt mapping을 이용하여 펼쳐진 (u,v) 좌표의 예

Pelt mapping은 스프링 모델을 적용한 파라미터라이제이션 기법을 말한다. (그림 22)는 3ds Max 8에서 제공하고 있는 pelt mapping의 예인데, 상단에 보인 캐릭터의 몸통 부분에 대한 파라미터라이제이션 과정을 보여준다. 몸통 메쉬의 경계(boundary) 버텍스에 가상의 스프링을 연결하고 그 주변에서 당겨주어 평면으로 편다. 메쉬의 비경계 선분(non-boundary edge)에도 역시 가상의 스프링을 설정하여, 가능한 한 원래

메쉬의 선분 길이로 돌아가게 하면, 파라미터라이제이션된 2차원 메쉬의 선분의 길이가 원래의 3차원 메쉬 선분 길이와 대체로 일정한 비율을 유지하게 된다.

## 5. 결 론

본 원고에서는 게임 그래픽 역사 및 기초 내용에 대해 살펴본 다음, 현재 게임개발자들이 가장 관심을 가지고 있는 부분을 그래픽 아티스트 내지 그래픽 툴이 어떻게 지원해 줄 수 있는지에 대해서 텍스처 매핑을 위주로 기술해 보았다. 그런데, 이러한 텍스처 매핑 기술 말고도, 게임 그래픽 분야에서는 수많은 기술적 진보들이 있어왔고, 이들은 프로그래밍 기술과 결합되어 현재 3차원 게임에서 구현되는 현란한 영상을 만들어 왔다.

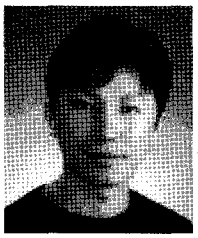
최근 Playstation3 및 Xbox360 게임 시연에서 증명되었듯이, 앞으로의 게임은 현재 수준을 훨씬 능가하는 엄청난 퀄리티의 영상을 제공해 줄 것이다. 실사영화와 같은 영상을 게임화면에 구현하는 것이 최종목표라고 봤을 때, 2차원 그래픽에서 3차원 그래픽으로 발전해 간 것은 그 긴 여정의 첫 발자국이었고, 현재의 3차원 그래픽 기술은 거기에서 몇 발자국 더 진보한 것일 뿐이라고 보면 될 것이다. 앞으로의 갈 길이 멀다. 본 원고가 독자들에게 지나온 여정과 현재 수준을 보여줌으로 해서, 앞으로의 먼 여정을 예측하게 하는데 도움이 되었으면 한다.

## 참고문헌

[1] M. Floater, Parameterization and smooth approximation of surface triangulations, Computer Aided Geometric Design, Vol.14, 1997, pp. 231~250.

[2] D. Piponi and G. Borshukov, Seamless Texture Mapping of Subdivision Surfaces by Model Pelting and Texture Blending, SIGGRAPH 2000.

## 저자약력

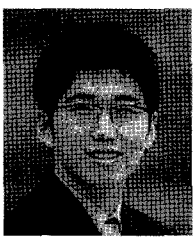


**김 일 혁**

1995년 순천대학교 토목공학과(학사)
1996년 (주)예일디자인아카데미 CAD학과 강사
1997년 (주)디웍스 그래픽 팀장
2000년 (주)게임스쿨 게임그래픽학과 강사
2002년 세종대학교 멀티미디어소프트웨어(석사)
2002년 (주)위메이드 엔터테인먼트 게임개발4부 캐릭터팀장
2003년~현재 Autodesk Korea Media and Entertainment 사업부 근무
e-mail : il-hyuk.kim@autodesk.com

**손 홍 정**

2004년 고려대학교 컴퓨터학과(학사)
2004년~현재 고려대학교 컴퓨터학과 석사과정
e-mail : hj79337@gmail.com



**정 택 상**

2005년 고려대학교 컴퓨터학과(학사)
2005년~현재 고려대학교 컴퓨터학과 석사과정
e-mail : nanocreation@gmail.com