3D 소프트웨어 교육방법에 대한 연구

A Study on the educational methode of 3D Software

최성원

동명정보대학교 정보조형학부

컴퓨터그래픽학과 전임강사

CHOI, Sung-Won

Dept. of Computer graphics, Tongmyong Univ.

● Keywords: 3D Software, Realism

1. 서론

본 연구의 목적은 現 대학교육에 있어서 어떻게 학생들에게 3D Software를 그 근본개념의 연장선 안에서 도구개념으로 이해시킬 수 있는가이며, 이는 3D Software를 한번도 대해본 적이 없는, 또는 자신이 사용하던 3D Software의 인터페이스가 완전히 변화하였을 경우, 학생들이 어떻게 가장 빠르고, 효율적으로 습득할 수 있게 하는가에 그 목적이 있다.

이를 위해서 본 연구는 어떤 일정한 Software를 기준으로 설명해 나아가는 것이 아니라, Software 일반에 대한 설명이기 때문에 가장 근본적인 내용, 즉 3D Software는 우리의 세상을 모방하여 영화와 같이 Computer에 재건하기 위해서 만들어진 Software인 만큼, 연구의 내용은 모든 Software의 가장 기초에 깔려 있는 '인간이라는 種으로서 세상이 어떻게 우리의 눈에 각인 되어지는가에 대한 논리적 사고'와 그에 해당하는 3D Software의 대개념들과의 상관관계를 그 내용으로 삼겠으며, 그 방법론으로는 우리의 눈에 각인 되어진 세상을 프로그래머들이 Software에 적용하여 사용자로 하여금 자신의 세계를 어떻게 표현하도록 제작하였는가에 대하여, 프로그래머의 입장에서 이해해야 한다는 것이 본 연구의 방법론이 되겠다.

2. 현실세계와 3D Software와의 관계

3차원의 의미는 무엇일까 ? 이러한 차원은 시각적 대상이 아니기 때문에, 그 의미를 알기 위해서는 해당되는 차원 내에 존재하는 대상들의 공통점을 찾아내어 유추할 수밖에 달리 도리가 없다. 그럼 3차원 공간 내에 존재하는 모든 대상들의 공통점은 과연 무엇일까 ? 우리의 눈에 보여지는 모든 대상들 즉 種 개념으로서 대상일반의 공통점을 가만히 생각해보면, 모든 대상은 부피를 가지고 있으며, 이러한 부피 때문에 우리 눈에 모든 대상이 각인 되어지는 것이다. 이러한 부피를 갖고 있지 않은 대상은 우리의 눈에는 보여지지 않는다. 따라서 현실세계의 3차원 공간은 부피가 면과 선을 포함하고 있기 때문에 그 이하의 차원을 포함함과 더불어 대상의 부피를 가능하게 하는 특징을 갖고 있다.

이러한 현실 공간세계를 그대로 옮겨 놓은 컴퓨터 내의 3차원 가상 세계는 수학(기하학)에 기초하여 만들어졌기 때문에, 그 이하의 모든 차원과 그에 해당하는 대상들 즉 점의 0차원, 선의 1차원 그리고 면의 2차원 모두를 포함하며, 마지막으로는 현실세계와 같은 부피에 의해서 완성되어진다. 그러나 대상의 부피성으로 대상성이 완성 되어지는 것은 아니다. 대상을 대상으로서 존재 하게끔 하는 대상임은 부피와 더불은 대상의 속성인 것이다. 즉 대상이 어떤 형태, 어떤 색상, 그리고 어떤 재질을 갖고 있는가에 따라 다른 대상과 구별되어진다. 이는 대상의 개별성을 근거 지우는 개별자의 특성인 것이다. 이러한 것을 프로그래머들은 3D Software에서 대상의 부피와 자신만의 고유한 모습을 재현시킬 수 있는 場으로서 Modeling 분야를, 그리고 속성인 색상과 재질을 만들 수 있는 場으로서는 Mapping 또는 Texturing이라는 분야를 만들어 사용자로 하여금 Software를 이용하여 현실의 Realism을 특징으로 갖는 가상의 대상들을 만들 수 있도록 설계하였다.

그럼, 이렇게 부피와 속성을 지닌 대상이 존재한다고 각인할 수 있는 모든 조건을 만족시킨다고 말할 수 있을까 ? 이는 현실세계의 빛 - 그것이 자연적인 또는 인위적인 빛이든 - 의 존재 없이는 어떠한 대상도 우리의 눈에 각인 되어질 수 없다. 따라서 대상의 모습은 부피와 속성을 지닌 대상에 투사되어진 빛과의 접촉점에 의해서 우리의 눈에 들어오는 대상의 모습이다. 따라서 빛은 대상의 모습을 우리의 눈에 비추게 하는 필수조건이다. 이러한 과정을 통해 우리의 눈에 각인 되는 대상의 모습이기에 아이러니컬하게 우리는 진정한 대상 자체의 모습을 알 수 없다는 결론이 나타난다. 이러한 현실세계의 상황을 이해한다면 얼마나 빛의 역할이 중요한지 알 수 있다. 이는 3D Software에서는 Light로 표현 되어진다. 이제 대상과 빛이 존재함으로서 비로소 대상은 우리의 눈에 각인 되어질 준비가 되어 있다. 그러나 우리의 눈이 존재하지 않는다면 대상은 대상으로서 우리에게 인식되어지지 못한다. 왜냐면 대상이란 우리에 대한 대상일 뿐 그 자체로서는 아무런 의미도 갖고 있지 않기 때문이다. 따라서 대상이 대상으로서 존재할 수 있는 조건은 우리의 눈이 필수조건인 것이다. 이는 영화의 Camera와 같으며, 3D Software에서도 역시 Camera로 표현된다. 이상의 모든 것은 컴퓨터내의 가상공간 내에서만 이루어질 수 있는 것이다. 여기에 시간의 의미가 개입하면서 운동(Animation)이 시작된다. 즉 운동은 시간이 전제되어야만 가능한 것이다. 이를 위해서 프로그래머들은 시간의 의미를 Frame으로서 사용자에게 제공해 주고 있다. 이로서 가상의 공간에서 한 걸음 더 나아가 가상의 시공간으로 의미가 확장되어진다.

그럼 우리가 상기에서 언급한 대상, 빛 그리고 카메라 중 운동이 불가능한 것이 있을까? 현실의 빛과 우리의 눈은 운동을 하기 때문에 현실을 모방한 빛과 카메라는 운동이 가능하다. 그럼 대상은 어떠한가? 상기에서 고정적인 대상이 있다고 말했다. 그러나 이것조차도 타자에 의해서 운동 가능하고, 또한 시간의 흐름에 따라 마모 내지는 소멸 가능한 대상이기 때문에, 그것이 비록 고정적 대상이라고 할지라도 그리고 대상의 속성이라고 할지라도 운동 가능하다.

이로서 우리는 시간의 흐름에 따라 변하지 않거나, 운동이 되어지지 않는 대상은 없다고 결론 지을 수 있다. 따라서 공간을 대상이 존재할 수 있는 근거라고 한다면, 시간은 대상이 운동할 수 있는 근거라고 할 수 있다. 이로서 대상은 우리의 눈에 각인될 수 있고, Animation이 가능한 모든 조건을 갖추었다. 부피와 속성을 가진 대상, 이것을 비추어줌과 동시에 대상을 우리의 눈에 반사 해주는 빛, 그리고 그러한 대상을 비로소 대상이게끔

하는 우리의 눈, 이 세 가지가 현실에서 우리가 대상을 각인하는 조건이며, 이를 모방하여 3D Software에서는 Modeling, Mapping 또는 Texturing, Light, Camera 그리고 Animation이라는 부분으로 구분해 놓았다. 그러나 문제는 현실의 세계에서는 우리의 눈이 대상을 볼 경우, 모든 대상이 실시간 적으로 아무런 지장 없이 각인 되어지는 반면, 컴퓨터에서는 현실과 같이 실시간적 각인이 불가능하다는 것이다. 따라서 이러한 것을 보완하기 위해서 만들어 놓은 부분이 바로 Rendering이다. 따라서 Rendering은 실세계에 비유를 한다면 비로소 대상이 부피와 속성을 가진 대상으로서, 바로 우리의 눈에 각인 되어 우리가 대상을 대상으로 인식하는 과정인 것이다. 이러한 Rendering은 인간의 시각적 착각에 의해서 보여지는 동영상 만화와 같은 시스템을 쓰고 있다. 즉 시간에 따른 각각의 Frame이 정지된 이미지로 나타나며, 이러한 일련의 이미지들의 연속성이 우리들의 눈에 움직이는 것처럼 보여지는 것이다.

3. 3D Software의 논리성

현실세계와의 비교로 3D Software는 Modeling, Texturing 또는 Mapping, Camera, Light, Animation, Rendering으로 분류되어져 있음을 살펴보았다. 그럼 이러한 구성을 사용자는 어떻게 쉽게 이해할 수 있을까 ? 이를 위해서 3D Software는 어떤 방법으로 설계되어 있을까 ?를 우선적으로 살펴보는 것이 문제 해결에 도움이 된다. 이는 인간의 대상 인식의 논리 구조를 따르고 있다. Kant의 인식론을 따르자면, 인간이 개별대상을 인식할 수 있는 까닭은 인식 내에 집이라고 함은 이러이러하다는 정의 - 개념이 내재한 까닭이라고 한다. 즉 種개념에서 시작하여 그 이하의 하부개념들이 차례로 파악되는 것이다. 이는 3D Software에서 뿐만 아니라 모든 Software에서 마찬가지이다. 즉 대개념 -> 중개념 -> 소개념으로 이루어져 있는 것이다. 여기서 워드프로세서의 예를 들면, 대개의 경우 File, Edit등과 같은 대개념 하에 각각 그 자신의 하부개념을 가지고 있다. 그리고 그 중의 몇 가지의 하부개념은 또한 그 자신만의 하부개념을 갖고 있다. 그럼 왜 'File' 이라고 불렀으며, 그 안에 왜 Open, Save와 같은 하부개념을 포함하고 있는 것일까 ? 파일은 현실에서 우리가 들고 다니는 서류철을 뜻하며, 여기서 우리는 서류를 꺼내어 보는데 이것이 바로 'Open'에 해당되어진다. 그리고 어떤 문서를 보고 난 후에 무엇을 적어 다시 서류철에 넣었을 때, 이것이 'Save'에 해당되어진다. 이처럼 컴퓨터에 있는 하나하나의 개념들은 제멋대로 만들어진 것이 아니라 인간의 인식 방법과 행태를 충분히 검토하고, 그 타당성이 입증 된 후 만들어진 것이며, 아이콘 역시 마찬가지 맥락이라고 할 수 있겠다. 이는 3D 소프트웨어에 있어서도 마찬가지이다. 비록 워드프로세서 보다 훨씬 더 복잡하지만, 전체적 구성은 인간의 이해에 기초한 대개념 -> 중개념 -> 소개념의 방식인 것이다. 따라서 사용자는 3D Software의 대개념에서부터 순차적으로 그들 개념을 이해해야 하며, 그 다음 그에 해당하는 단어의 개념을 잘 이해한다면, 어느 정도 Software의 기능들을 개념적으로 이해한다고 말할 수 있다. 왜냐면 단어는 설명의 개념적 표현이기 때문에, 이것이 사용자와 Software 프로그래머간의 첫 번째 대화창이기 때문이다. 그 후, 구체적으로 그 기능들을 적용시키면, 대부분의 경우 대화창 하단에 그 기능에 대한 설명 내지는 지시사항 또는 사용자가 잘못 적용한 것에 대한 설명이 나타난다. 이러한 메시지는 사용자의 이해를 돕기 위한 것이기 때문에 새로운 Software를 사용하는 사람들은 간과해서는 안될 부분이다. 따라서 사용자는 처음 대하는 3D Software에서 가장 먼저 해야 할 일은 어디에서 modeling을 하고, 어디서 animation을 하며, 어디서 mapping을 하는가 등의 대개념의 위치들을 알아두어야 하며, 그리고 그에 해당하는 하부개념이 어디에서 작용하는지 또한 어느 부분에서 작용하는지에 대해서 알아야 하며, 그와 더불어 단어의 의미 파악이 선행되어져야 할 것이다.

4. 결론

이상에서 살펴 본 것처럼 3D Software는 Modeling, Mapping(Texturing), Animation, Lighting, Camera 그리고 Rendering으로 분류되어짐을 알았다. 이것이 근래에 와서는 Particles, Dynamics, Kinetics 및 여러 가지 특수효과등이 첨부되어 3D Software의 기능이 더욱 세분화, 복잡화되어 간다고 할 수 있겠다. 이러한 것들은 또 다른 설명이 필요한 것들이나, 전체적인 3D Software의 기본 구성은 상기에서 언급한 틀에서 크게 벗어나지 않는다. 따라서 이상과 같은 3D Software 일반에 대한 이해, 그리고 대개념에서 하부개념으로의 순차적 이해과정이 이루어지고, 이러한 이해 하에서 구체적 기능을 적용 시켜본다면, 3D Software들에 대한 사용자의 적응 능력은 매우 유연해질 것이다.