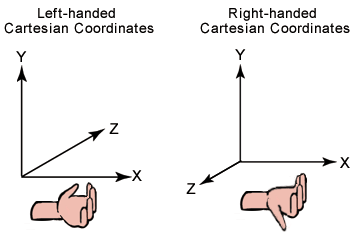
왼손 vs 오른손 좌표계, 행우선 vs 열우선 표현법.

2차원 이상의 카테시안 좌표계는 방향성에 따라 두가지로 구분할 수 있는데, 왼손좌표계(left-handed)와 오른손좌표계(right-handed system)가 그것이다. 이 둘 사이에 대수적 차이는 존재하지 않고, 온전한 동치를 이루는데, 우리가 해야할 일은 어떤 좌표계를 사용할지 선택하는 것 뿐이다. 그것마저도 오른쪽좌표계를 사용하는 수학적 관례를 따른다면 자연스럽게 해결된다.



그런데 유달리 3D 프로그래밍에 있어서 왼손/오른손좌표계가 이슈화되는 것은 이 두 좌표계가 여러 어플리케이션 사이에 혼재되어 사용되고 있기 때문이다. 즉, 협업하는 어플리케이션들 사이의 좌표계를 꼼꼼히 따져줄 필요성이 생기는 것이다. 예를 들면, Max 혹은 Maya 같은 저작도구와 그로부터 제작된 모델을 사용하는 프로그램 사이에 좌표계가 동일한지 확인하고, 다르다면 적절한 조치를 취해주어야 한다.

그 적절한 조치란, 좌표계를 혼용할 경우 나타나는 문제들, 즉 z축이 미러링(mirroring)되고 폴리곤의 노멀이 반전되는 것을 역으로 되돌려 놓기만 하면 되는 것이다[[1]](http://msdn.microsoft.com/en-us/library/bb204853.aspx).

이렇게 간단한 개념에 대해 수많은 오해와 논쟁이 계속 되는 이유는 무엇일까? 나는 두 가지 이유에서 그렇다고 생각한다.

첫째는 이유를 찾으려는 시도 때문일 것이다. 예를 들어, 왼손좌표계만을 사용하던 사람들이 오른손좌표계를 사용하는 환경을 보면, 생소한 얼굴로 왜 그런 좌표계를 사용하고 있는지 이유부터 찾으려고 한다. 정작 자기 자신도 왜 왼손좌표계를 사용하고 있는지 정확히 정의하지 못하면서 말이다. 실상은 아무런 이유가 없다.

또, 자신이 만들고 있는 어플리케이션이 어떤 좌표계를 사용할지는 전적으로 자기 자신에게 달려있음에도 사용하고 있는 라이브러리로부터 제약당하고 있다고 착각하는 경향이 있다.

DirectX와 OpenGL의 코어라이브러리들은 좌표계에 대한 아무런 정의를 내리고 있지 않다. 다만 그것을 지원하는 유틸리티 라이브러리(D3DX, GLU)가 뷰 매트릭스 혹은 프로젝션 매트릭스를 계산할 때 특정 좌표계를 사용하는 것 뿐이다. 그나마도 DirectX는 두 좌표계를 위한 함수를 각각 제공하고 있다. 선택은 전적으로 사용자에게 달려있는 것이다.

어느 좌표계를 선택하건 그것은 분명 기호의 문제다. 그래도 DirectX가 수학적 관례인 오른손좌표계를 포기하고 왼손좌표계를 기본(default)으로 채택한 이유를 굳이 찾자면 무엇일까? 명확히 알려진 바는 없지만(당연하게도) 추측컨데 [렌더링 파이프라인](http://en.wikipedia.org/wiki/Graphics_pipeline) 전과정에 걸쳐 일관성 있는 좌표계를 유지하고 싶었기 때문이 아닐까 싶다. 일반적으로 프로젝션 변환(transformation) 과정에서 far plane의 z값이 near plane의 z값보다 큰 값을 가지도록 하기 위해서 왼손좌표계로의 변환이 일어난다. 가까운 값이 먼 값보다 수치적으로 작아야 직관적으로 느껴지기 때문이다. 이런 이유로 처음부터 왼손좌표계를 사용함으로서 z축이 반전되는 상황을 없애고 싶었을지도 모를일이다.

두번째 이유는 컴퓨터가 행렬을 저장하는 두가지 방식인 [행우선(row major)과 열우선(column major)방식](http://en.wikipedia.org/wiki/Row_major)이 왼손 또는 오른손 좌표계와 연관이 있다고 오해하는 것이다. OpenGL이 내부적으로 행렬을 곱할 때(glRotate, glMultMatrixf 등) 열우선방식을 사용하는 것은 사실이다. 그리고 DirectX가 제공하는 행렬관련 함수들이 행우선방식을 사용하는 것도 사실이다. 그러나 이것이 오른손좌표계에서는 열우선방식을 사용해야 하고, 왼손좌표계에서는 행우선방식을 사용해야함을 의미하지는 않는다.

수학적 관례에 따르면 열우선방식이 자연스러운 선택일 것이다. 대부분의 수학책에서 벡터를 4x1행렬로 표현하며, 행렬과의 곱셈에서 후위곱(post-multiplication)방식을 사용하기 때문이다.

그러나 C/C++을 다루는 입장에서 자연스러운 행렬표현법은 행우선방식이다. 행렬을 C/C++에서의 2차원배열로 정의했을 때, 첫째 인덱스로 행을 두번째 인덱스로 열을 가리킬 수 있기 때문이다. 또한 연속적인 행렬을 곱할 때 전위곱(pre-multiplication)방식을 사용하므로, 왼쪽에서 오른쪽으로 순서대로 변환이 적용되기 때문에 변환의 순서를 읽거나 쓸때 직관적인 장점이 있다.

요점은 약간의 대수적 고려만 있으면, 어떤 그래픽 라이브러리에 대해서도 원하는 행렬표현법을 선택해서 사용할 수 있으며, 어떤 것을 선택할지는 개인 취향의 문제라는 것이다.  
행우선/열우선 방식에 대한 참고자료는 다음과 같다.

* <http://steve.hollasch.net/cgindex/math/matrix/column-vec.html>  
  OpenGL 명세서를 작성한 Mark Segal이 유즈넷에 쓴 17년 전의 글을 통해 이것에 관한 논란(?)의 역사가 얼마나 오래되었는지 알 수 있다.
* <http://www.3dbuzz.com/vbforum/showthread.php?t=171267>  
  좀 지루하긴 하지만, 좋은 동영상 강의 자료.

개인적으로 오른손좌표계와 행우선방식을 선호하는데, [XNA framework](http://www.xna.com/), [WPF](http://en.wikipedia.org/wiki/Windows_Presentation_Foundation) 디자이너들도 같은 생각을 한 것 같다.