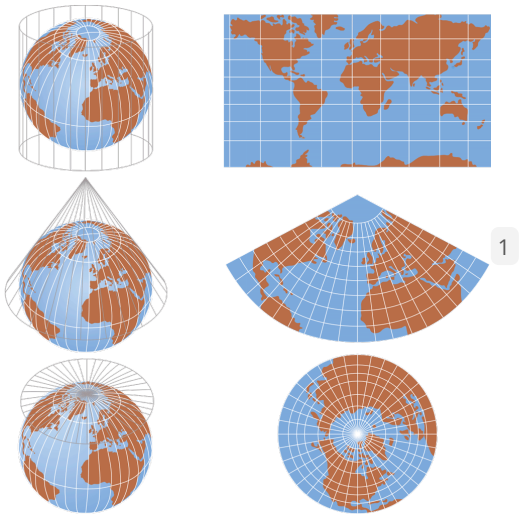
3차원 공간 화면에 투영하기

활동 배경

3D 게임을 하며, 게임 속 3차원 공간을 2차원 공간에 투영하는 원리가 무엇인지 궁금증을 가지게 되었다. 그런데 이번에 기하 과목을 수강하며 벡터와 정사영이라는 개념을 접하고 공간을 계산하는 방법을 배우면서 위의 궁금증을 직접 해결해 보기로 했다.

활용 사례

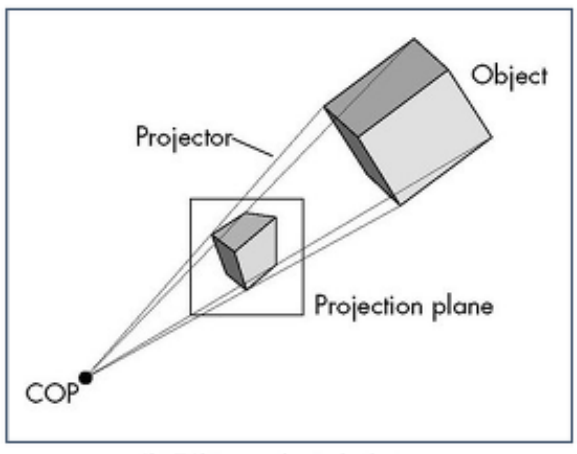
3차원의 공간을 2차원인 화면에 투영하는 기술을 다양한 분야에서 활용된다. 앞서 활동 동기가 된 3D 게임 뿐만 아니라, 영화에 사용되는 컴퓨터 그래픽(CG), 물리 시뮬레이션, 입체감을 부여하는 디자인 기법 등에 사용된다.



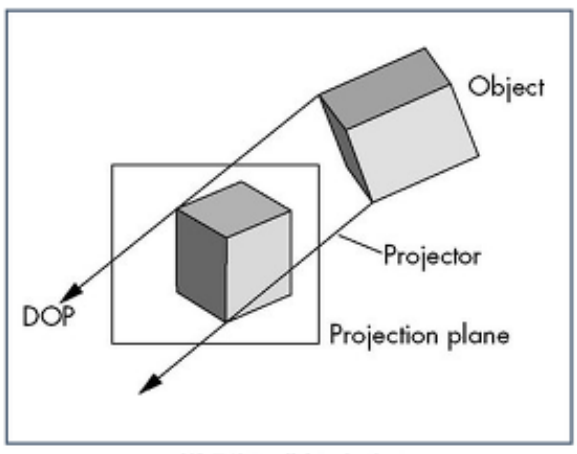
또한 세계지도 제작에도 다양한 투영변환 기법이 사용된다. 두 번째와 세 번째가 각각 원근 투영과 평면 투영이다.

원리

투영이란[2] 크게 투영참조점(projection reference point, PRP), 그리고 투영선(projector), 투영 평면(projection plane, PP)에 의해 정의된다. 투영참조점은 투영이 전달되는 좌표를, 투영선은 투영 대상으로부터 투영참조점을 연결한 선을, 투영 평면은 투영선과의 교점을 통해 결과를 도출하는 스크린과 같은 역할을 가진다.

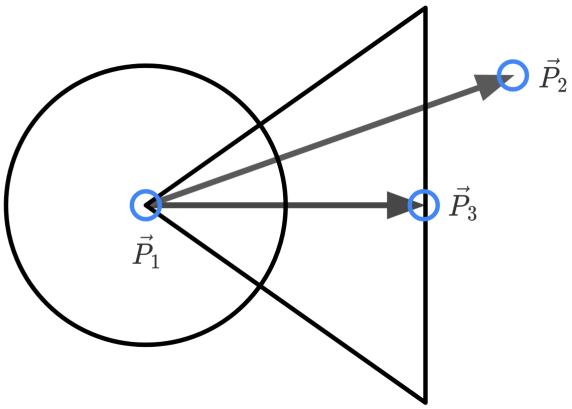


원근투영(Perspective Projection)



평행 투영(Parallel Projecting)

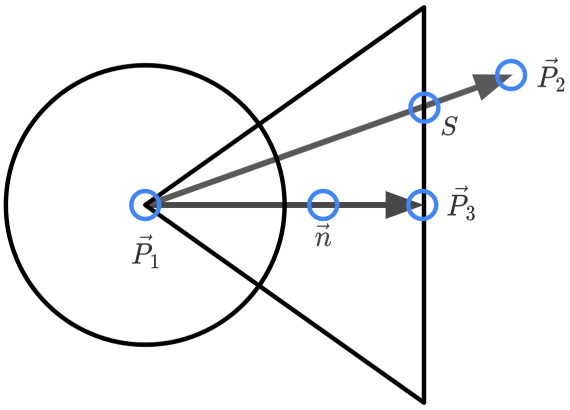
투영의 두 가지 대표적 예시로 원근투영과 평행투영이 있다. 원근 투영은 인간의 시야와 유사한 시야를 가지고, 평면 투영은 물체의 형태를 변화시키지 않고 전달한다. 원근 투영은 투영선과 투영 평면 사이의 교점을 구해야 한다는 귀찮은 점이 있다. 반면, 평면 투영은 투영 평면에 대한 정사영이므로 상대적으로 구현하기 쉽다.



위의 그림은 3차원에서 원근투영 계산이 이루어지는 모습의 단면과 그 계산에 필요한 값들이다. 그림에서 세로선은 화면에 출력할 스크린 부분이고, `P_1`, `P_2`, `P_3`은 각각 `카메라의 좌표`, `투영대상의 좌표`, `스크린 중앙의 좌표`이다.

구현

`n`은 `P_1`을 지나고 평면인 스크린의 법선벡터이다. 이 값들을 적절히 계산해서 물체가 화면상에서 갖는 좌표를 구한다. `S`는 스크린 위에 나타나는 투영 대상의 좌표이다.



기본적인 구조는 카메라가 물체를 인식한다는 것이다. 또한 모든 점들을 공간벡터로 생각하고, 카메라 좌표에 대한 상대좌표만을 사용해 계산을 진행했다. 계산을 진행하면서 스크린과 `P_1`, `P_2`를 이은 직선을 정의하여 둘 사이의 교점을 구했다. 계산 과정은 아래와 같다.

> 법선벡터가 $\vec{n}$ 이고, $\vec{P}_3$을 지나는 평면 $\vec{P}$ 는 다음과 같이 정의된다.
> $\vec{n} \cdot (\vec{P} - \vec{P}_3) = 0$
>
> $\vec{P}_1, \vec{P}_2$ 를 지나는 직선 $\vec{P}$ 는 다음과 같이 정의된다.
> $\vec{P} = \vec{P}_1 + u \cdot (\vec{P}_2 - \vec{P}_1)$
>
> 평면의 식에 직선의 식을 대입하면

$$\vec{n}\cdot(\vec{P_1}+u(\vec{P_2}-\vec{P_1})-\vec{P_3}),\ \vec{n}\cdot u(\vec{P_2}-\vec{P_1})=\vec{n}(\vec{P_3}-\vec{P_1}),$$
$$u=\frac{\vec{n}(\vec{P_3}-\vec{P_1})}{\vec{n}(\vec{P_2}-\vec{P_1})}$$

여기서 u값의 범위는 (0, 1]로, 이 범위를 벗어나면 화면 뒤편의 좌표로, 출력 범위에서 제외된다. 또한, `n`은 아래와 같이 간편하게 정의할 수 있고, u 또한 계산할 수 있다. u의 값은 컴퓨터를 사용해 계산할 것이다.

$$\begin{aligned}\vec{n}=&\vec{P_3}-\vec{P_1}\\ u=&\frac{\vec{n}(\vec{P_3}-\vec{P_1})}{\vec{n}(\vec{P_2}-\vec{P_1})}=\frac{(\vec{P_3}-\vec{P_1})\cdot(\vec{P_3}-\vec{P_1})}{(\vec{P_3}-\vec{P_1})\cdot(\vec{P_2}-\vec{P_1})}\\ =&\frac{|\vec{P_1}|^2+|\vec{P_3}|^2-2\vec{P_1}\cdot\vec{P_3}}{|P_1|^2-\vec{P_1}\cdot(\vec{P_2}+\vec{P_3})+\vec{P_2}\cdot\vec{P_3}}\end{aligned}$$

점 S의 좌표 또한 아래의 방법으로 구할 수 있다.

$$S=u\cdot\hat{P_2}$$

이제 `s`의 값을 알기 때문에 스크린 위에서의 좌표 또한 알 수 있다. 하지만 스크린 위의 좌표를 알기 위해서는 `s`의 값을 한 단계 손봐줄 필요가 있다. 그 방법으로는 첫 번째로 세 개의 점과의 거리를 통해 위치를 특정하는 것, 두 번째로 기저를 변환시키는 방법이 있다. 다만 이 부분은 아직 완성되지 않아서 추가적인 개념의 학습이 필요하다. 기저의 변환[3]은 아래와 같이 나타내 지는데, 이를 한쪽 방향으로 변환하는 방식을 일반화하지 못해 적용할 수 없었다.

> 2차원 실수 공간 $\mathbb{R}^2$ 의 두 개의 기저 $\mathcal{B}=\{v_1,v_2\},\mathcal{C}=w_1,w_2$ 가 있다.
> 또한 $k_1v_1+k_2v_2=l_1w_1+l_2w_2$ 로 나타낼 수 있다.
> 좌표 벡터와 행렬을 이용하면 아래의 식을 구해낼 수 있다.
> $$[v]\mathcal{B}=\begin{bmatrix}k_1\\k_2\end{bmatrix}=\begin{bmatrix}al_1+cl_2\\bl_1+dl_2\end{bmatrix}=\begin{bmatrix}a&c\\b&d\end{bmatrix}\begin{bmatrix}l_1\\l_2\end{bmatrix}=\begin{bmatrix}a&c\\b&d\end{bmatrix}[v]\mathcal{C}$$
> 이때, 벡터 v의 $\mathcal{B}$표현을 $\mathcal{C}$ 표현으로 바꾸어주는 행렬 $\begin{bmatrix}a&c\\b&d\end{bmatrix}$ 을 기저변환행렬$(transitionmatrix)$이라 부른다.

활동 소감

이번 활동을 통해 배운 것도 많고 생각한 것도 많지만, 무엇보다 좋았던 것은 내가 수업시간에 학습한 내용을 그 순간에만 남겨두지 않고 실용적으로 사용할 방법을 찾았다는 사실이었다. 특히 내가 관심있는 분야의 내용에 대해 추가로 학습하고 목표를 달성하는 과정이어서 그런지 동기부여와 결과 모두 좋게 나왔던 것 같다. 또한 이와 관련해서 물체의 회전과 같이 비슷한 분야에 대해 더욱 학습해 보고 싶고, 추후 고급수학을 수강하면서 행렬 개념을 배우면 회전변환을 활용하여 더욱 적합한 방법을 찾는 것이 또 다른 목표이다. 다만, 이 프로젝트를 완결내지 못한 것이 아쉽게 다가온다. 원래의 계획은 3차원 공간을 투영하는 방법을 파악하고 이를 프로그램을 통해 직접 구현하는 것이었는데, 이중 식 구성과 프로그램 구현을 끝마치지 못했다. 이 또한 방학이나 이후의 시간을 투자해 종결지을 생각이다.

---

1. https://www.blinklearning.com/coursePlayer/clases2.php?idclase=47993669&idcurso=888959 ↩
2. https://robodream.tistory.com/217 ↩
3. https://angeloyeo.github.io/2020/12/07/change_of_basis.html ↩