UNITY

Intermediate

Lerp, SmoothDamp

Lerp란?

특정 오브젝트를 특정 좌표로, 원하는 시간동안 이동하게 하고싶을 때가 있습니다. 물론 좌표로 직접 옮겨줘도 되지만, "일정 시간" 만큼 이동시켜 주는 것은 생각보다 어려운 일이죠. 바로 이 때 Lerp가 사용됩니다.

Lerp(start, end, t)

에서, start에는 시작값이, end에는 목표값이, t에는 0부터 1 사이의 실수가 들어가는데 여기서 이 실수는 시작값부터 목표값까지의 이동한%를 의미합니다. 그러니까 start가 0이고, end가 100일때 t가 0.5라면 50, 0.1이라면 10, 1이라면 100을 반환합니다.

이 때, Lerp는 Vector3.Lerp 도 있고, Mathf.Lerp 도 있고, 매우 다양한 타입에 Lerp가 존재하기 때문에 자신이 필요한 타입을 사용하도록 합시다. Lerp는 다른 말로 **선형 보간**이라고 할 수 있습니다. `start + (end - start) * t` 라는 식을 사용합니다.

SLerp란?

SLerp는 선형 보간인 Lerp와 다르게, 구면 선형 보간을 합니다. 그래서 시작점 A부터 끝점 B까지 간다는 것은 동일하지만, A와 B를 잇는 포물선을 만든 후, 그 포물선 위의 위치를 반환한다고 생각하면 편합니다.

SLerp(start, end, t)

사용법은 Lerp와 동일합니다. 만약 실제 이동이 궁금하다면 아래의 영상을 참고해주세요!

https://youtu.be/Bki5I-dMj0c

Lerp 예제

아래와 같이 작성한 후에, Inspector 창에서 원하는 좌표를 넣어주면 5초에 걸쳐서 해당 좌표로 이동하게 됩니다. 시작 후 지난 시간 / 지속시간 한 값을 넣어줌으로써 N초에 필요한 위치를 알 수 있게 되는거죠!

```
public Vector3 endPos;
Unity 메시지 | 참조 0개
void Start()
{
    StartCoroutine(MoveTo(endPos, 5f));
}

참조 1개
IEnumerator MoveTo(Vector3 dest, float duration)
{
    Vector3 start = transform.position;
    float elapsedTime = 0;
    while (elapsedTime <= duration)
    {
        transform.position = Vector3.Lerp(start, dest, elapsedTime / duration);
        elapsedTime += Time.deltaTime;
        yield return null;
    }
    transform.position = dest;
}
```

Lerp 응용

그런데 Lerp는 Easing을 하기 위해 사용될 수도 있습니다. Easing이라는 것은, 부드러운 움직임을 뜻하는데, 예시로 다음과 같은 코드가 있습니다.

```
public Vector3 endPos;
Unity 메시지 | 참조 0개
void Update()
{
    transform.position = Vector3.Lerp(transform.position, endPos, Time.deltaTime);
}
```

이 경우 현재 위치 기준으로 목표 위치까지의 이동이니, 처음에 1/10만큼 움직였다면 그다음에는 남은거리인 9/10의 /10만큼... 이런식으로 반복할수록 기준치도 작아져서 결과적으로 목표에 도달할수록 부드럽게 느려지는 것을 구현할 수 있습니다.

하지만 이 경우 함수의 극한처럼 거의 절대 목표에 도달할 수가 없게 되고, 끝에서 너무 느려진다는 문제가 있어서 이 경우에는 SmoothDamp를 사용합니다.

SmoothDamp란?

SmoothDamp는 시간에 따라 원하는 목표를 향해 **점차적**으로 값을 바꿔주는 것으로, 두 값 사이의 부드러운 이동을 가능하게 해줍니다.

SmoothDamp(current, target, ref curVelocity, smoothTime, maxSpeed = Mathf.Infinity, deltaTime = Time.deltaTime)

여기서 ref란 Reference의 약자로, 값을 복사하는 것이 아니라 참조하겠다는 뜻입니다. C#의 문법에 해당하기 때문에 궁금하시다면 out과 같이 참고해보세요.

current는 현재 위치, target는 목표 위치이며 curVelocity는 현재 속도, smoothTime는 목적지 도달까지의 대략적인 시간이며 maxSpeed는 최대속도를 지정해줄 수 있게 하고 deltaTime은 이 함수가 마지막으로 호출된 후 경과한 시간입니다.

SmoothDamp 예제

아래와 같이 해주면 끝나게 됩니다. 정확히 설정한 시간만큼 걸리지는 않는다는 것을 알 수 있죠.

```
public Vector3 endPos, velocity;
public float smoothTime = 3f;
Unity 메시지 | 참조 0개
void Update()
{
    transform.position = Vector3.SmoothDamp(transform.position, endPos, ref velocity, smoothTime);
}
```