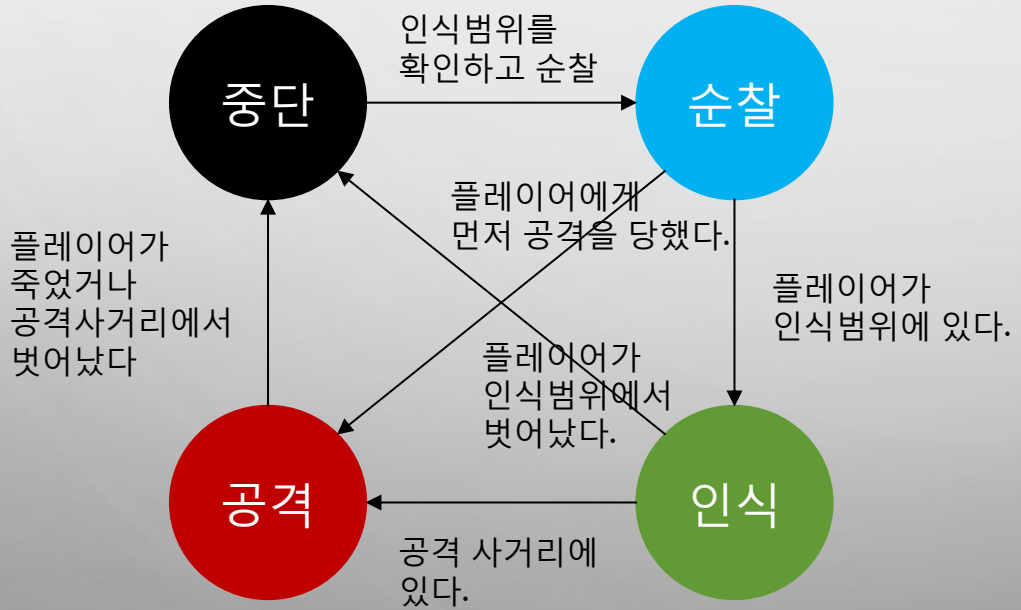
UNITY 3D
-CHAPTER6-

SOULSEEK

FSM(FINITE STATE MACHINE)

- 인공지능 기법 중 하나, 유한한 개수의 상태를 가지는 추상기계이다.
- 현재 상태로부터 가능한 전이 상태와 이러한 전이를 유발하는 조건의 집합으로 정의한다.
- 프로그래밍에서는 실행 시점에서 현재의 상태에 기반해 객체의 코드 일부를 선택적으로 수행함으로써 객체의 행동을 제어하는 용도.(ex 적의 플레이어 수색 AI)
- If ~else와 true, false가 혼잡하게 섞이는 것을 피하는 방법이다.
- 상태를 전이해서 동작시키는 함수가 필요.
- Unity의 Animator시스템에서 이러한 것을 볼 수 있다.



COROUTINE

- Multi Thread처럼 사용하기 위해서 쓴다.
- Yield라는 키워드를 사용한다.
- StartCoroutine(IEnumerator형 함수);로 실행, StopAllCoroutine()으로 종료

```
void Fade()
{
	for(float f = 1f; f >= 0f; f -= 0.1f)
	{
		Color c = GetComponent<Renderer>().material.color;
		c.a = f;
		GetComponent<Renderer>().Material.color = c;
	}
}
```

```
IEnumerator Fade()
{
	for(float f = 1f; f >= 0f; f -= 0.1f)
	{
		Color c = GetComponent<Renderer>().material.color;
		c.a = f;
		GetComponent<Renderer>().Material.color = c;
		yield return null;
	}
```

FSM과 COROUTINE

```
public class EnemyAI : MonoBehaviour
{
    public enum State
    {
        PATROL,
        TRACE,
        ATTACK,
        DIE
    }

    public State state = State.PATROL;

    void OnEnable()
    {
        //코루틴 실행부분
        StartCoroutine(CheckState());
    }
}
```

```
IEnumerator CheckState()
{
    while(!isDie)
    {
        if(state == State.DIE)
            yield break;

        float dist = Vector3.Distance(playerTr.position,
                                      enemyTr.position);

        if(dist <= sttackDist)
        {
            state = State.ATTACK;
        }
                        .
                        .
                        .
        yield return new WaitForSeconds(0.3f);
    }
}
```

SENDMESSAGE & DELEGATE

```
public class EnemyAI : MonoBehaviour
{
    public void OnPlayerDie()
    {
        moveAgent.Stop();
        enemyFire.isFire = false;
        StopAllCoroutines();
            .
            .
            .
    }
}
```

SendMessage 함수는 특정 게임오브젝트에 포함된 스크립트를 하나씩 검색해 호출하려는 함수가 있다면 실행하라고 메시지를 전달한다. 스크립트가 존재 하지 않다면 오류를 발생시키고 수가 많을 때는 비효율적이다.

```
Public class Damage : MonoBehaviour
{
    void PlayerDie()
    {
        GameObject[] enemies
        = GameObject.FindGameObjectsWithTag(enemyTag);

        for(int I = 0; I < enemies.Length; i++)
        {
            //다른 곳에서 함수 호출 방법.
            enemies[i].SendMessage("OnPlayerDie",
                SendMessageOptions.DontRequireReceiver);
        }
    }
}
```

SENDMESSAGE & DELEGATE

```
Public class Damage : MonoBehaviour
{
    //델리게이트 와 이벤트 선언.
    public delegate void PlayerDieHandler();
    public static event PlayerDieHandler OnPlayerDie;

    void PlayerDie()
    {
        //연결 된 이벤트  실행
        OnPlayerDie();
    }
}
```

이벤트 드리븐을 이용해서 직접적으로 함수를 연결 시켜 준다.

```
Public class EnemyAI : MonoBehaviour
{
    void OnEnable()
    {
        //이벤트 연결
        Damage.OnPlayerDie
                += this.OnPlayerDie;
    }

    void OnDisable()
    {
        //이벤트 해제
        Damage.OnPlayerDie
                -= this.OnPlayerDie;
    }

    public void OnPlayerDie()
    {
        moveAgent.Stop();
        StopAllCoroutines();
    }
}
```