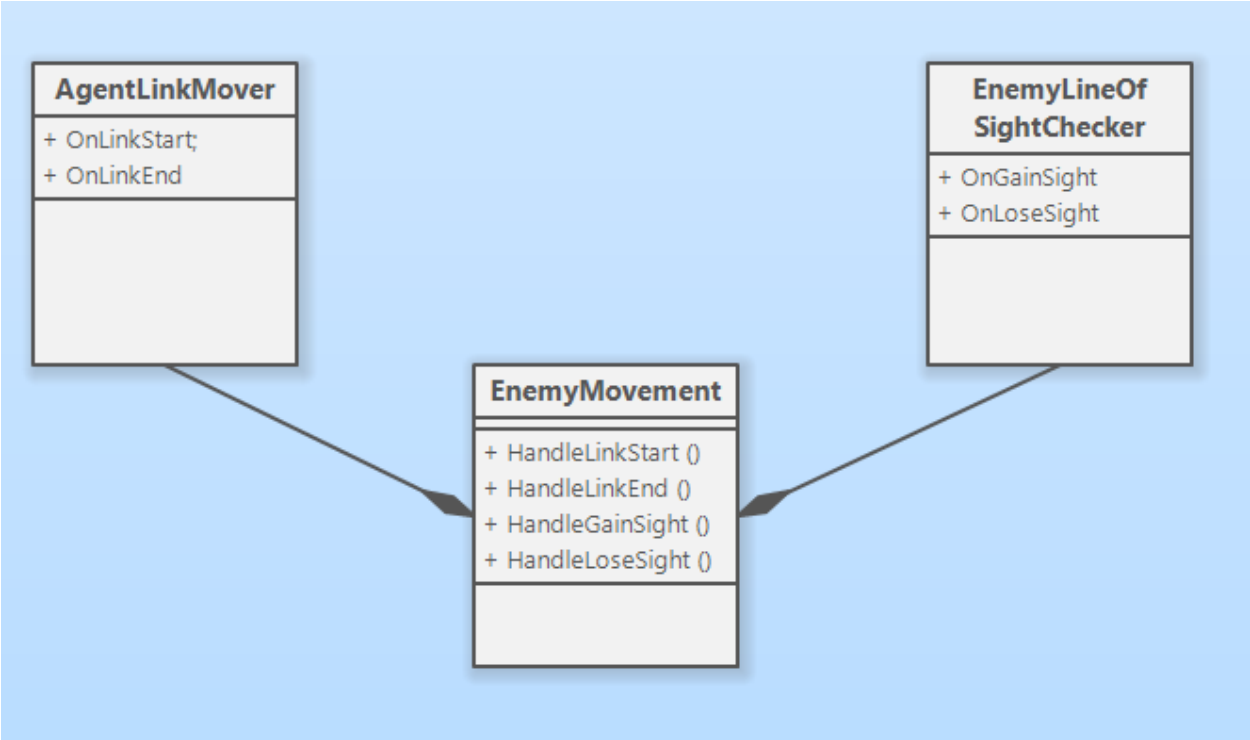
Посередник - це поведінковий патерн проєктування, який дає змогу зменшити пов'язаність безлічі класів між собою, завдяки переміщенню цих зв'язків до одного класу-посередника.

В моєму проекті класом посередником виступає EnemyMovement. Цей клас зв’язує різні компоненти поведінки ворога в єдине ціле (хоча, не тільки він, тому що цей клас відповідає тільки за переміщення та стан противника).

Наприклад є інший клас AgentLinkMover, цей клас відповідає за “стрибки” противника між платформами, а також є компонент який відповідає за анімації. І клас EnemyMovement об’єднує ці класи. Тобто коли AgentLinkMover дає знак, що стрибає, EnemyMovement запускає анімації в компоненті Аніматора. Іншими словами реалізує потрібний функціонал, який не потрібен в класі AgentLinkMover, а також прибирає залежність цього класа від аніматора.

Ще один приклад EnemyLineOfSightChecker, перевіряє чи є противник для нашого противника (тобто гравець) в зоні видимості. Якщо так, то він запускає метод, який реалізується в EnemyMovement, аналогічно коли гравця не має в зоні видимості. Тобто EnemyMovement поєднує зміну стана об’єкта (противника) з класом EnemyLineOfSightChecker.



Вказівники на класи.

```
public class EnemyMovement : MonoBehaviour
{

    public GameObject Player;
    public EnemyLineOfSightChecker LineOfSightChecker;
    [SerializeField]
    private Animator animator = null;
    public NavMeshTriangulation Triangulation = new NavMeshTriangulation();
    [Tooltip("Update of state")]
    public float UpdateRate = 0.1f;
    [HideInInspector]
    public NavMeshAgent Agent;
    private AgentLinkMover linkMover;
    [SerializeField]
    public EnemyState DefaultState;
    private EnemyState _state;
```

В методі Awake ми присвоюємо потрібний функціонал.

```
private void Awake()
{
    DefaultPositon = transform.position;
    Agent = GetComponent<NavMeshAgent>();
    linkMover = GetComponent<AgentLinkMover>();
    linkMover.OnLinkStart += HandleLinkStart;
    linkMover.OnLinkEnd += HandleLinkEnd;
    if (LineOfSightChecker == null)
        LineOfSightChecker = GetComponent<EnemyLineOfSightChecker>();
    LineOfSightChecker.OnGainSight += HandleGainSight;
    LineOfSightChecker.OnLoseSight += HandleLoseSight;
    OnStateChange += HandleStateChange;
    HandleStateChange(State, DefaultState);


}
```

Цей патерн допомагає уникнути “спагеті-коду”, що при великих розмірах проекта полегшує його підтримку. Також добре підходить за для вирішення проблем поєднання, коли потрібно взяти функціонал з різних класів, але не потрібно додавати додаткову залежність цим класам

Скрипти можете знайти за шляхом:

Assets/Scripts/Enemies/Base/EnemyMovement.cs

Assets\Scripts\Enemies\Base\EnemyLineOfSightChecker.cs

Assets\Scripts\Enemies\Base\AgentLinkMover.cs