2021.09.17

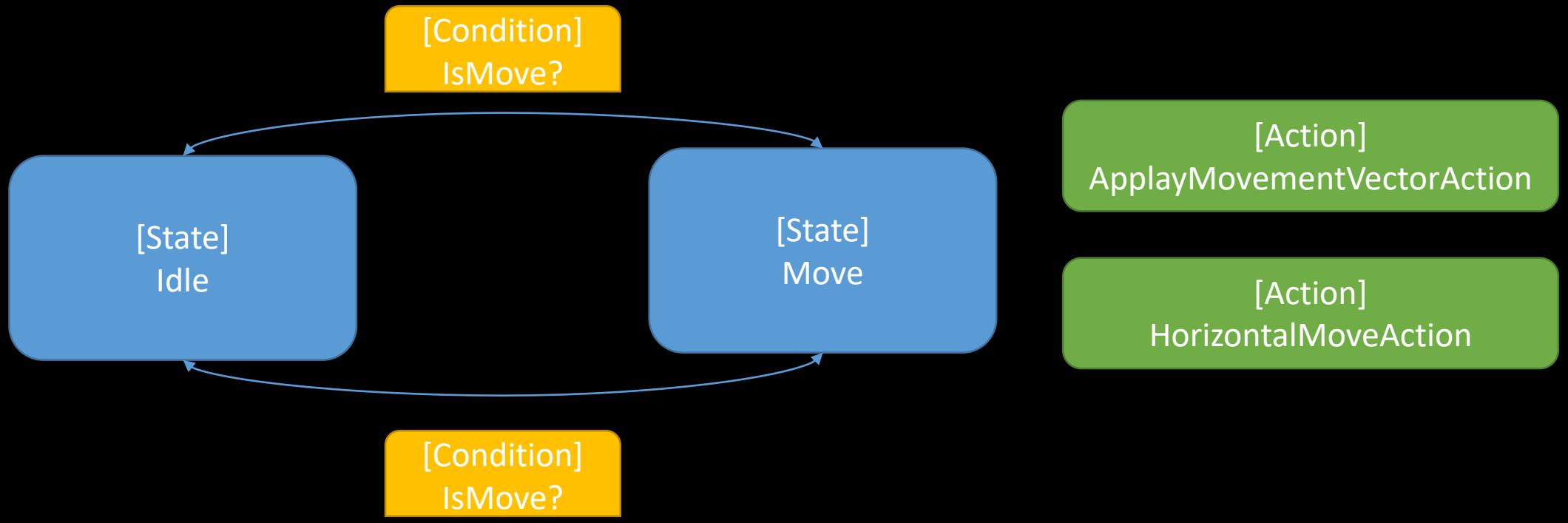
Unity FSM

작성자 : 김은규

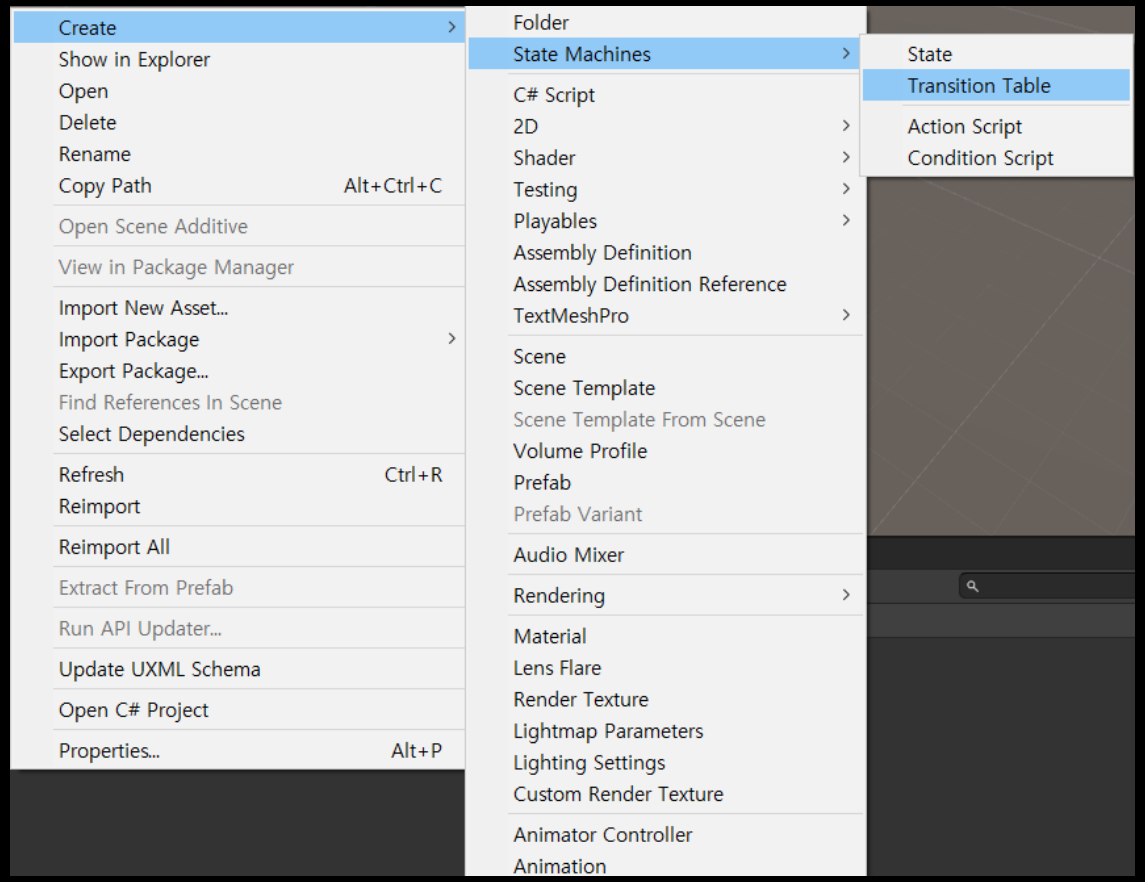
움직임을 구현하기 위한 단계

- 0. StateMachine Table 생성
- 1. 상태 정의(Idle, Move) [State]
- 2. 키 입력 코드 작성 [Player]
- 3. 상태의 전이 조건 정의 [Condition]
- 4. 키 입력 코드 움직임을 위한 Vector3로 변환 [Action]
- 5. 실제 움직임 정의 [Action]
- 6. State에 Actio SO 등록
- 7. StateMachine Table에 State, Condition, Action 적용

StateMachine 구성

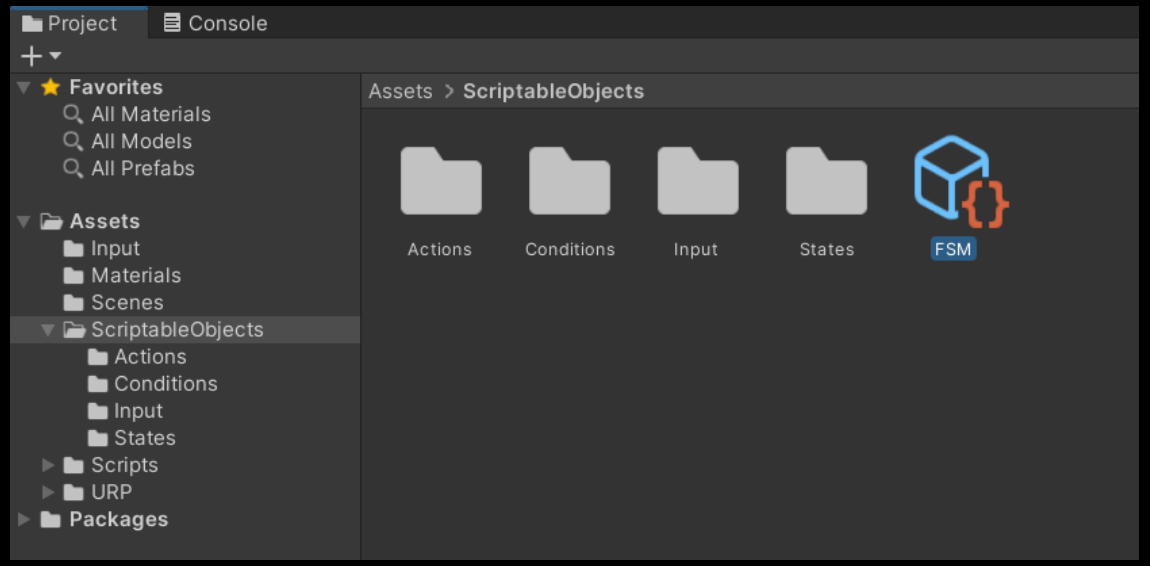


0.StateMachine Table



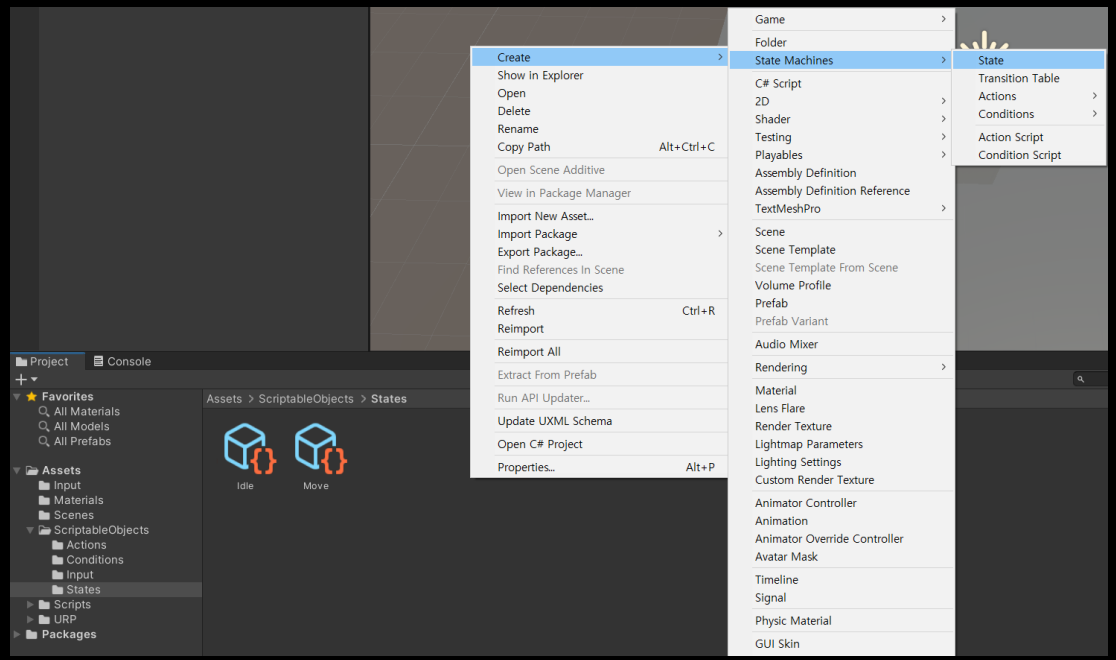
- 상태 전이를 관리할 테이블 생성

0.StateMachine Table



- ScriptableObject로 생성된 FSM

1. State 생성



- FSM 상태 생성
- Idle, Move

2. 키 입력 코드 작성

```
Unity 스크립트 | 참조 6개
public class Player : MonoBehaviour
{
    public InputReader inputReader;

    public Vector2 previousMove;
    public Vector3 inputVector;
    public Vector3 moveVector;

    Unity 메시지 | 참조 0개
    private void Awake()
    {
        inputVector = new Vector3();
    }

    Unity 메시지 | 참조 0개
    private void OnEnable()
    {
        inputReader.moveEvent += ApplayPreviousMove;
    }

    Unity 메시지 | 참조 0개
    private void OnDisable()
    {
        inputReader.moveEvent -= ApplayPreviousMove;
    }

    참조 2개
    private void ApplayPreviousMove(Vector2 input)
    {
        previousMove = input;
    }

    // Update is called once per frame
    Unity 메시지 | 참조 0개
    void Update()
    {
        inputVector = new Vector3(previousMove.x, 0, previousMove.y);
    }
}
```

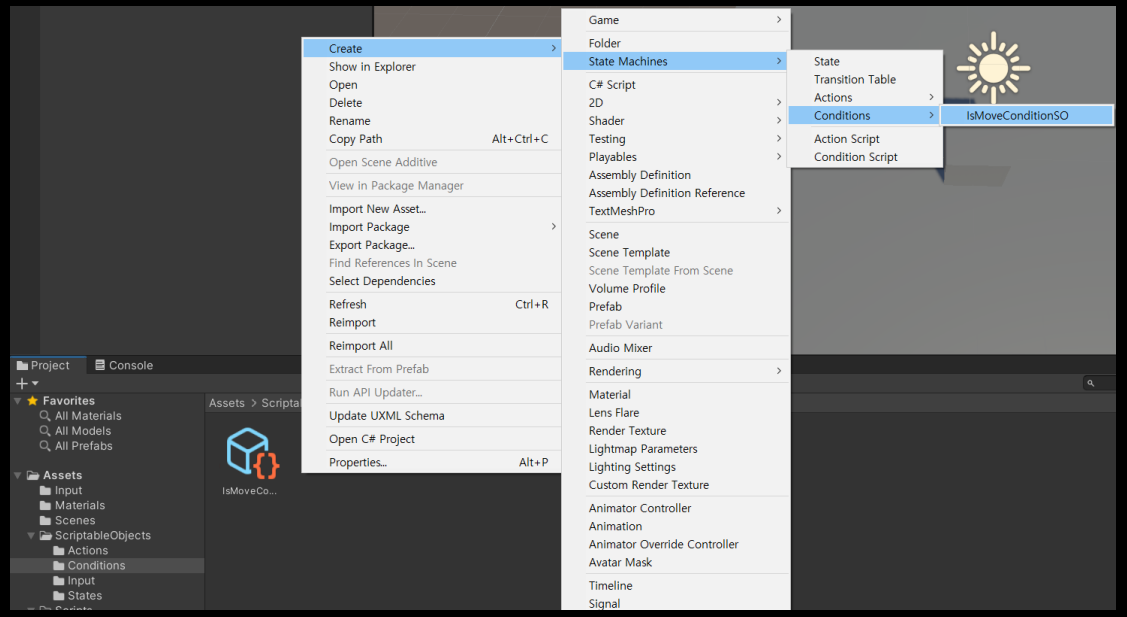
- inputReader의 moveEvent가 input값 전달

3. 상태의 전이 조건 정의 코드

```
6
7   [CreateAssetMenu(menuName = "State Machines/Conditions/IsMoveConditionSO")]
    Unity 스크립트|참조 2개
8   public class IsMoveConditionSO : StateConditionSO<IsMoveCondition>
9   {
10      public float tresholde = 0.02f;
11  }
12
    참조 1개
13  public class IsMoveCondition : Condition
14  {
15      private Player player;
        참조 1개
16      private IsMoveConditionSO originSO => (IsMoveConditionSO)base.OriginSO;
        참조 2개
17      public override void Awake(StateMachine stateMachine)
18      {
19          player = stateMachine.GetComponent<Player>();
20      }
        참조 4개
21      protected override bool Statement()
22      {
23          Vector3 movementVector = player.inputVector;
24          return movementVector.sqrMagnitude > originSO.tresholde;
25      }
26
        참조 6개
27      public override void OnStateExit()
28      {
29          player.moveVector = Vector3.zero;
30      }
31  }
32
```

- Tresholde 값 이상일 경우 상태 전이 IsMoveCondition : true

3. 상태의 전이 조건 정의 SO



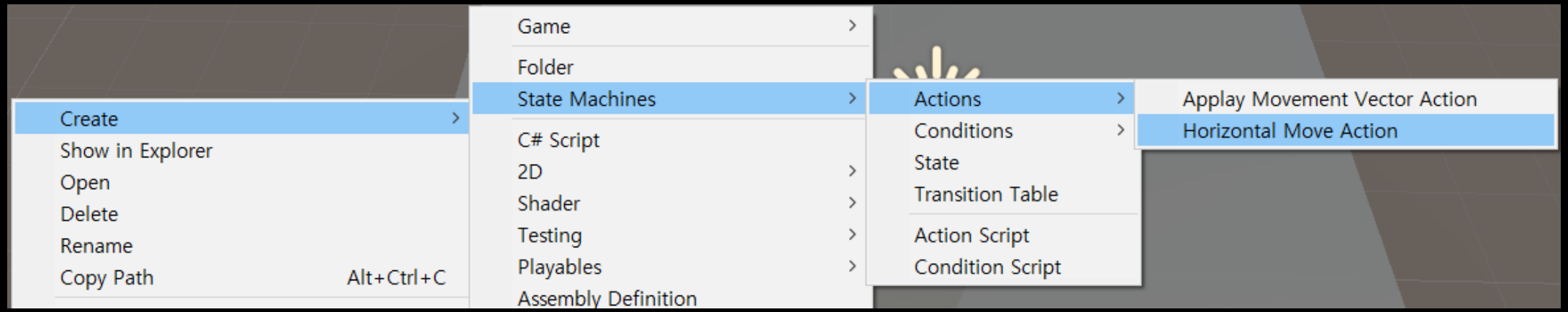
- StateMachines>Conditions> IsMoveConditionSO
- 로 생성

4. 키 입력 코드 움직임을 위한 Vector3로 변환 코드

```
using UnityEngine;
using UOP1.StateMachine;
using UOP1.StateMachine.ScriptableObjects;

[CreateAssetMenu(fileName = "HorizontalMoveAction", menuName = "State Machines/Actions/Horizontal Move Action")]
Unity 스크립트 | 참조 2개
public class HorizontalMoveActionSO : StateActionSO<HorizontalMoveAction>
{
    //protected override StateAction CreateAction() => new HorizontalMoveAction();
    public float speed = 8f;
}

참조 1개
public class HorizontalMoveAction : StateAction
{
    private Player player;
    참조 2개
    private HorizontalMoveActionSO origin => (HorizontalMoveActionSO)base.OriginSO;

    참조 3개
    public override void Awake(StateMachine stateMachine)
    {
        player = stateMachine.GetComponent<Player>();
    }

    참조 3개
    public override void OnUpdate()
    {
        player.moveVector.x = player.inputVector.x * origin.speed;
        player.moveVector.y = 0;
        player.moveVector.z = player.inputVector.z * origin.speed;
    }

    // public override void OnStateEnter()
    // {
    // }

    // public override void OnStateExit()
    // {
    // }
}
```

- Play가 움직이기 위한 moveVector 연산

4. 키 입력 코드 움직임을 위한 Vector3로 변환 SO 생성



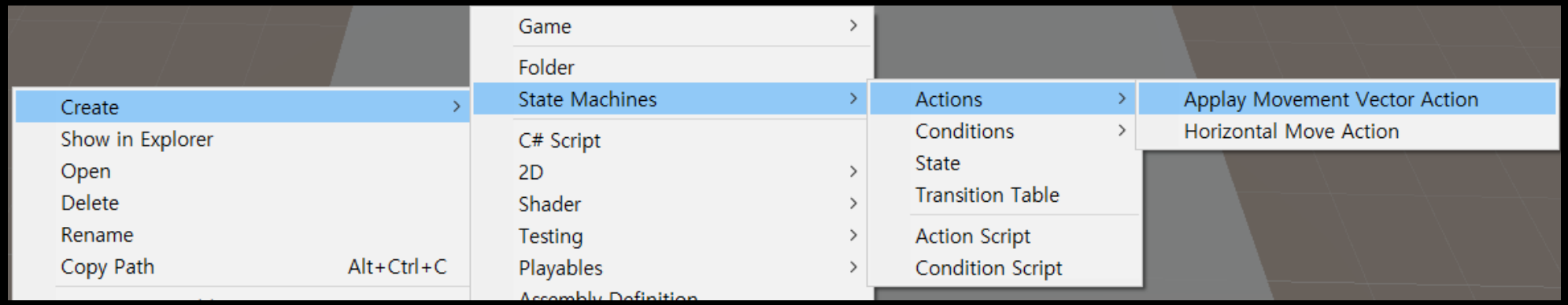
- Horizontal Move Action SO 생성

5. 실제 움직임 정의 코드

```
using UnityEngine;
using UOP1.StateMachine;
using UOP1.StateMachine.ScriptableObjects;

[CreateAssetMenu(fileName = "ApplayMovementVectorAction", menuName = "State Machines/Actions/Applay Movement Vector Action")]
Unity 스크립트 | 참조 0개
public class ApplayMovementVectorActionSO : StateActionSO
{
    참조 3개
    protected override StateAction CreateAction() => new ApplayMovementVectorAction();
}

참조 1개
public class ApplayMovementVectorAction : StateAction
{
    private Player player;
    private CharacterController characterController;
    참조 3개
    public override void Awake(StateMachine stateMachine)
    {
        player = stateMachine.GetComponent<Player>();
        characterController = player.GetComponent<CharacterController>();
    }

    참조 3개
    public override void OnUpdate()
    {
        characterController.Move(player.moveVector * Time.deltaTime);
    }

    // public override void OnStateEnter()
    // {
    // }

    // public override void OnStateExit()
    // {
    // }
}
```

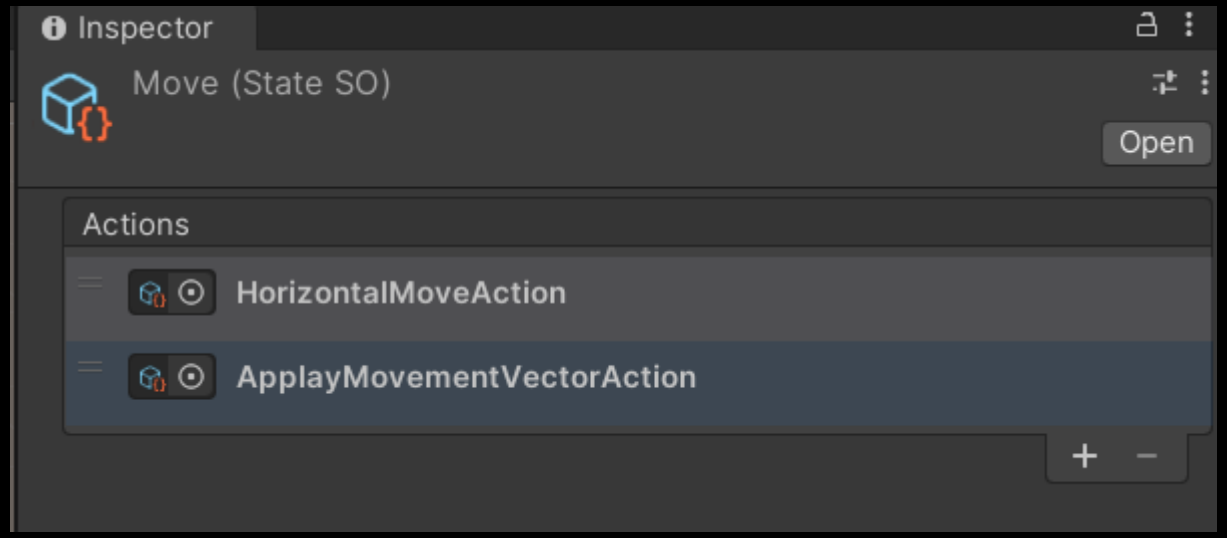
- moveVector로 캐릭터 실제 움직임 구현

5. 실제 움직임 정의 SO 생성



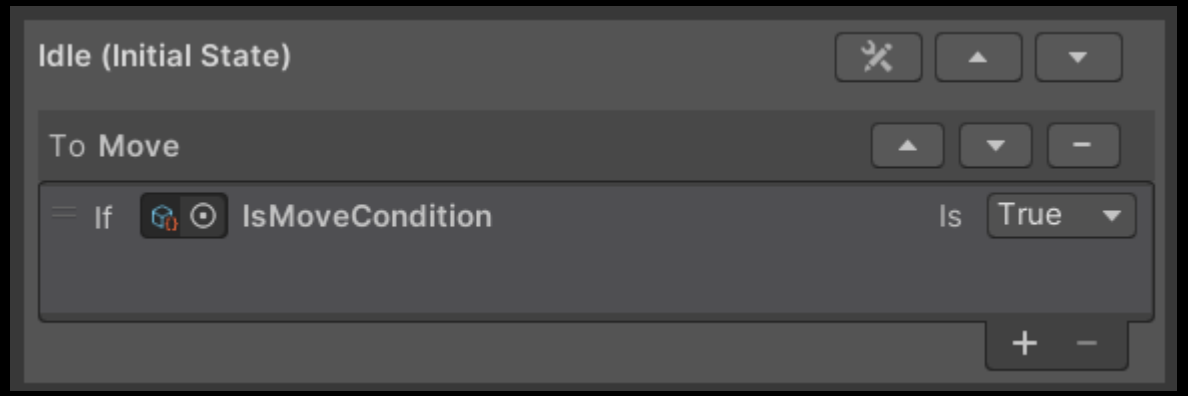
- Applay Movement Vector Action SO 생성

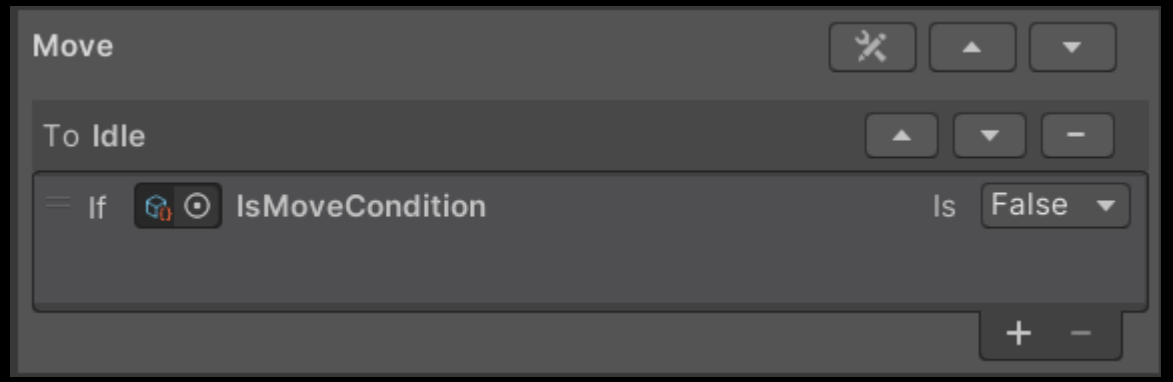
6. State에 Actio SO 등록



- Move상태 일때 이루어지느 Action 등록

7. StateMachine Table에 State, Condition, Action 적용





- Idle -> Move 전이 Condition 등록
- Move -> Idle 전이 Condition 등록