Document

1. Introduction

This folder contains the state machine code used in Canopy Quest for controlling enemies.

2. EnemyState

The `EnemyState` class serves as a base class for defining different states for an enemy in the game. States represent the various behaviors or conditions that an enemy can exhibit during gameplay. This class provides methods to handle state transitions and defines placeholders for specific state-related actions.

2.1 `EnterState`

```
public EnemyState(EnemyNew enemy, EnemyStateMachine enemyStateMachine)
```

- **Description:** Initializes the `enemy` and `enemyStateMachine` fields with the provided references.

2.2 `EnterState`

- **Description:** Placeholder method called when entering this state. Subclasses can override this method to define specific actions to be taken when entering the state.

2.3 `ExistState`

- **Description:** Placeholder method called when exiting this state. Subclasses can override this method to define specific actions to be taken when exiting the state.

2.4 `FrameUpdate`

- **Description:** Placeholder method called during the frame update. Subclasses can override this method to define specific actions to be taken during each frame update.

2.5 `PhysicsUpdate`

- **Description:** Placeholder method called during the physics update. Subclasses can override this method to define specific physics-related actions to be taken during each physics update.

2.6 `AnimationTriggerEvent`

- **Description:** Placeholder method called when an animation trigger event occurs. Subclasses can override this method to handle animation trigger events specific to the state.

3. EnemyStateMachine

The `EnemyStateMachine` class is responsible for managing the state of an enemy in the game. It keeps track of the current state and provides methods to initialize the state machine with a starting state and change the current state to a new one.

3.1 `Initialize`

```
public void Initialize(EnemyState startState)
```

- **Description:** Initializes the state machine with a starting state.
- **Parameters:** `startState` (EnemyState): The initial state of the enemy.

3.2 `ChangeState`

```
public void ChangeState(EnemyState newState)
```

- **Description:** Changes the current state of the enemy to a new state.
- **Parameters:** `newState` (EnemyState): The new state to transition to.

4. EnemyAttackState (as an example)

The `EnemyAttackState` class represents the attack state of an enemy character. It inherits from the `EnemyState` base class and overrides methods for entering the state, exiting the state, frame updates, and physics updates. This class serves as a bridge between the general state management system (`EnemyState`) and the specific attack behavior implemented in a scriptable object (`EnemyAttackSOBase`).

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EnemyAttackState : EnemyState
{
    public EnemyAttackState(EnemyNew enemy, EnemyStateMachine enemyStateMachine)
: base(enemy, enemyStateMachine)
    {
    }

    public override void AnimationTriggerEvent(EnemyNew.AnimationTriggerType
triggerType)
    {
        base.AnimationTriggerEvent(triggerType);
        enemy.enemyAttackBaseInstace.DoAnimationTriggerEventLogic(triggerType);
    }

    public override void EnterState()
    {
        base.EnterState();
        enemy.enemyAttackBaseInstace.DoEnterLogic();
    }

    public override void ExistState()
    {
        base.ExistState();
        enemy.enemyAttackBaseInstace.DoExitLogic();
```

```
    }

    public override void FrameUpdate()
    {
        base.FrameUpdate();
        enemy.enemyAttackBaseInstace.DoFrameUpdateLogic();
    }

    public override void PhysicsUpdate()
    {
        base.PhysicsUpdate();
        enemy.enemyAttackBaseInstace.DoPhysicsLogic();
    }
}
```