Quick Write-Up:

The entire point of this framework is to allow developers to create their own states (and transitions) without having to write the code for the state machine. Since I’m planning to use interfaces for the states (and transitions), developers should be able to write code for their own states (and transitions) and allowing them to use their own states (and transitions). The reason for allowing developers to code their own states (and transitions) is because the state machine will trigger methods within the States and Transitions. Writing your own States and Transitions, allows to use generic Stats and Transitions most of the time while using specialty States and Transitions when needed.

States will have an OnEnter and an OnExit method. This way, if there’s any code that needs to run when you enter into a state, then that code can be executed when you enter said state from the state itself. The same can be said about the OnExit method. Transitions will have an OnExecute method that will execute between the OnExit method of the Source State and before the OnEnter method of the Destination State.

Item Overview:

Objects:

* State
* Transition
* StateMachine
* StateMachineStack

Interface: IState

* ID : int
* Name : string
* Description : string
* ExecuteOnLoop : bool
* OnEnter()
* OnExit()

Interface: ITransition

* ID : int
* Name : string
* Event : string
* Description : string
* SourceState : IState
* DestinationState : IState
* OnExecute()

Interface: IStateMachine

* States: Collection<IState>
* Transitions : Collection<ITransition>
* InitialState : IState
* CurrentState : IState
* RegisterEvent(string event)

Class : StateMachine ← IStateMachine

* States : Collection<IState>
* Transitions : Collection<ITransition>
* InitialState : IState
* CurrentState : IState
* RegisterEvent(string event)

Class : StateMachineStack ← IStateMachine

* StateMachines : Stack<IStateMachine>
* CurrentStateMachine : IStateMachine