C# Reference

Steelbreeze.Behaviour Namespace

The Steelbreeze.Behaviour namespace provides the classes and interfaces that support UML style hierarchical finite state machines.

The Steelbreeze.Behaviour namespace is in the Steelbreeze.Behaviour assembly (in Steelbreeze.Behaviour.dll).

There is no StateMachine class in the namespace as either Region or OrthogonalState can be used as the top-level of the state machine.

This IState interface provides an interface for the state of the state machine (what states are currently active, etc); by providing an interface only it is up to the developer to implement their desired transaction, serialisation and thread management.

Classes

Class	Description
CompositeState	A state that contains child states and pseudo states.
Element	Any state, pseudo state or region within a state machine hierarchy.
FinalState	A state that cannot be transitioned out of.
OrthogonalState	A state than contains child regions.
PseudoState	A transient state with behaviour defined by its ‘kind’.
Region	A container of states and pseudo states.
SimpleState	A leaf-level state within a state machine hierarchy.
Transition<T>	A typed transition from a state to another state or pseudo state.
Transition	An untyped transition from a state or pseudo state to another state or pseudo state tested for upon entry to the source state or pseudo state.
Transition.Else	A catch-all completion transition from junction or completion pseudo states used where no other transitions guards evaluate true.

Interfaces

Interface	Description
IState	An interface from which to implement a container for the state of the state machine hierarchy

Enumerations

Euneration	Description
PseudoStateKind	Defines the behaviour of an individual PseudoState.

CompositeState Class

A state that contains child states and pseudo states.

Inheritance Hierarchy

System.Object

  Steelbreeze.Behaviour.Element

    Steelbreeze.Behaviour.SimpleState

      Steelbreeze.Behaviour.CompositeState

Syntax

```
public class CompositeState : SimpleState
```

The CompositeState type exposes the following members:

Constructors

Name	Description
CompositeState(String, Region)	Initialises a new instance of the CompositeState class with the given name and owning Region.
CompositeState(String, CompositeState)	Initialises a new instance of the CompositeState class with the given name and owning CompositeState.

Properties

Name	Description
Name	The name of the CompositeState. From Element.
Owner	The owning Element of the CompositeState. From Element.
QualifiedName	The fully qualified name of the CompositeState. From Element.

Methods

Name	Description
IsComplete(IState)	Determines if a CompositeState is complete by testing if the current active child is a FinalState.
OnEnter	Protected method allowing sub-classes to override entry behaviour. From SimpleState.
OnExit	Protected method allowing sub-classes to override exit behaviour. From SimpleState.
Process(IState, Object)	Attempts to process a message. This will first test if this state has any valid transitions for the type of the message and where the guard evaluates true; if none are found, it will delegate the message to the currently active child state.

Events

Name	Description
Entry	Occurs when a CompositeState is entered.
Exit	Occurs when a CompositeState is exited.

Remarks

A CompositeState is the most straightforward way to add hierarchy to a state machine. It fulfils all the semantics of the base SimpleState and augments it with child states and pseudo states.

The implementation of an IState interface will remember the last known child state for a composite state.

Element Class

Any state, pseudo state or region within a state machine hierarchy.

Inheritance Hierarchy

System.Object

  Steelbreeze.Behaviour.Element

Syntax

```
public abstract class Element
```

The Element type exposes the following members:

Properties

Name	Description
Name	The name of the Element.
Owner	The owning Element of the Element.
QualifiedName	The fully qualified name of the Element.

Remarks

An Element is an abstraction over members of the state machine hierarchy.

FinalState Class

A state that cannot be transitioned out of.

Inheritance Hierarchy

System.Object

  Steelbreeze.Behaviour.Element

    Steelbreeze.Behaviour.SimpleState

      Steelbreeze.Behaviour.FinalState

Syntax

```
public sealed class FinalState : SimpleState
```

The FinalState type exposes the following members:

Constructors

Name	Description
FinalState(String, Region)	Initialises a new instance of the FinalState class with the given name and owning Region.
FinalState(String, CompositeState)	Initialises a new instance of the FinalState class with the given name and owning CompositeState.

Properties

Name	Description
Name	The name of the CompositeState. From Element.
Owner	The owning Element of the CompositeState. From Element.
QualifiedName	The fully qualified name of the CompositeState. From Element.

Methods

Name	Description
IsComplete(IState)	Determines if a Final is complete. From SimpleState.
Process(IState, Object)	Attempts to process a message. As it is not possible to add transitions to FinalStates, this will always have no effect and return false.

Remarks

A FinalState marks is owning Region or CompositeState as being 'complete' and as such, are a critical mechanism for determining if the lifecycle of the entire object under state management is complete.

Completion transitions from CompositeStates will only be evaluated if the CompositeStates current active child is a FinalState.

Completion transitions from OrthogonalStates will only be evaluated if the current active child state of all the OrthogonalStates child Regions are FinalStates.

OrthogonalState Class

A state that contains child Regions.

Inheritance Hierarchy

System.Object

Steelbreeze.Behaviour.Element

Steelbreeze.Behaviour.SimpleState

Steelbreeze.Behaviour.OrthogonalState

Syntax

```
public class OrthogonalState : SimpleState
```

The OrthogonalState type exposes the following members:

Constructors

Name	Description
OrthogonalState(String, Region)	Initialises a new instance of the OrthogonalState class with the given name and owning Region.
CompositeState(String, CompositeState)	Initialises a new instance of the OrthogonalState class with the given name and owning CompositeState.

Properties

Name	Description
Name	The name of the OrthogonalState. From Element.
Owner	The owning Element of the OrthogonalState. From Element.
QualifiedName	The fully qualified name of the OrthogonalState. From Element.

Methods

Name	Description
IsComplete(IState)	Determines if an OrthogonalState is complete by testing if the current active child of all child Regions are FinalStates.
OnEnter	Protected method allowing sub-classes to override entry behaviour. From SimpleState.
OnExit	Protected method allowing sub-classes to override exit behaviour. From SimpleState.
Process(IState, Object)	Attempts to process a message. This will first test if this state has any valid transitions for the type of the message and where the guard evaluates true; if none are found, it will delegate the message to all child Regions.

Events

Name	Description
Entry	Occurs when an OrthogonalState is entered.
Exit	Occurs when a OrthogonalState is exited.

Remarks

An OrthogonalState add hierarchy to a state machine through multiple child Regions. It fulfils all the semantics of the base SimpleState and augments it with child states and pseudo states.

The child Regions of an OrthogonalState are meant to be independent of each other; any messages and actions delegated to an OrthogonalState are further delegated to all child Regions as required.

Therefore, when OrthogonalStates are used, the current state of a state machine can actually have multiple values as each child Region maintains its own current state.

An OrthogonalState is recommended as the top-level of a state machine where orthogonal child regions are required at the top-level of the state machine; otherwise use a Region.

PseudoState Class

A transient state with behaviour defined by its ‘kind’.

Inheritance Hierarchy

System.Object

Steelbreeze.Behaviour.Element

Steelbreeze.Behaviour.PseudoState

Syntax

```
public class PseudoState : Element
```

The PseudoState type exposes the following members:

Constructors

Name	Description
PseudoState(String, PseudoStateKind, Region)	Initialises a new instance of the PseudoState class with the given name, kind and owning Region.
PseudoState(String, PseudoStateKind, CompositeState)	Initialises a new instance of the PseudoState class with the given name, kind and owning CompositeState.

Properties

Name	Description
Kind	The kind of the PseudoState.
Name	The name of the PseudoState. From Element.
Owner	The owning Element of the PseudoState. From Element.
QualifiedName	The fully qualified name of the PseudoState. From Element.

Remarks

A PseudoState is a transient state in a state machine without entry or exit behaviour; upon entry, completion transitions will be evaluated to determine the next state of the state machine.

Each Region or CompositeState must have a single initial PseudoState who’s kind is Initial, DeepHistory or ShallowHistory; this is used to control the initial starting position when entering the Region or CompositeState.

Other kinds of PseudoStates are used as branching mechanisms to provide composite transitions enabling complex logic and behaviour between states.

Region Class

A container of states and pseudo states..

Inheritance Hierarchy

System.Object

Steelbreeze.Behaviour.Element

Steelbreeze.Behaviour.Region

Syntax

```
public class Region : Element
```

The Region type exposes the following members:

Constructors

Name	Description
CompositeState(String, OrthogonalState)	Initialises a new instance of the CompositeState class with the given name and owning Region.

Properties

Name	Description
Name	The name of the Region. From Element.
Owner	The owning Element of the Region. From Element.
QualifiedName	The fully qualified name of the Region. From Element.

Methods

Name	Description
Initialise(IState)	Initialises a Region to its initial state when used as a top-level state machine.
IsComplete(IState)	Determines if a Region is complete by testing if the current active child is a FinalState.
Process(IState, Object)	Attempts to process a message. This will delegate the message to the currently active child state.

Remarks

A Region is a container of states and pseudo states, usually as a child of an OrthogonalState.

A Region is recommended as the top-level of a state machine in most circumstances (only where the top level required orthogonal regions would an OrthogonalState be a better choice).

SimpleState Class

A leaf-level state within a state machine hierarchy.

Inheritance Hierarchy

System.Object

Steelbreeze.Behaviour.Element

Steelbreeze.Behaviour.SimpleState

Syntax

```
public class SimpleState : Element
```

The CompositeState type exposes the following members:

Constructors

Name	Description
SimpleState(String, Region)	Initialises a new instance of the SimpleState class with the given name and owning Region.
SimpleState(String, CompositeState)	Initialises a new instance of the SimpleState class with the given name and owning CompositeState.

Properties

Name	Description
Name	The name of the SimpleState. From Element.
Owner	The owning Element of the SimpleState. From Element.
QualifiedName	The fully qualified name of the SimpleState. From Element.

Methods

Name	Description
IsComplete(IState)	Determines if a SimpleState is complete; always true for SimpleStates.
OnEnter	Protected method allowing sub-classes to override entry behaviour.
OnExit	Protected method allowing sub-classes to override exit behaviour.
Process(IState, Object)	Attempts to process a message. This will test if this state has any valid transitions for the type of the message and where the guard evaluates true. If found, it will perform the transition.

Events

Name	Description
Entry	Occurs when a SimpleState is entered.
Exit	Occurs when a SimpleState is exited.

Remarks

A SimpleState is the most straightforward state in a state machine. It defines a condition during the life of an object under state management. Upon entering the state, the Entry event is called and completion transitions are tested for; upon exiting the state the Exit event is called.

Transition<T> Class

A typed transition from a state to another state or pseudo state.

Inheritance Hierarchy

System.Object

Steelbreeze.Behaviour.Transition<T>

Syntax

```
public class Transition<T> where T : class
```

The Transition type exposes the following members:

Constructors

Name	Description
Transition(SimpleState, PseudoState, Func<T,Boolean>)	Initialises a new instance of the Transition class from a SimpleState to a PseudoState with an optional guard condition.
Transition(SimpleState, SimpleState, Func<T,Boolean>)	Initialises a new instance of the Transition class from a SimpleState to a SimpleState with an optional guard condition.
Transition(SimpleState, Func<T,Boolean>)	Initialises a new instance of the Transition class creating an internal transition for a SimpleState with a guard condition.

Methods

Name	Description
OnEffect	Protected method allowing sub-classes to override transition behaviour.

Events

Name	Description
Effect	Occurs when a Transition<T> is traversed.

Remarks

A Transition<T> is a typed transition from a SimpleState (or any class derived from SimpleState) to another SimpleState or PseudoState.

When a message is passed to a state for evaluation, firstly, its type is checked against the types of the transitions from that state, then for those of matching type the guard condition is evaluated. If a single matching transition is found it is traversed and a state transition occurs. If multiple are found, an exception is thrown as the machine is deemed to be malformed.

When exiting a state, any child structure is first exited (if the state is a CompositeState or OrthogonalState), then its Exit event is called.

When entering a state, its Entry event is called then any child structure is entered (if the state is a CompositeState or OrthogonalState).

If the target of the transition is a PseudoState (or another SimpleState or derivative that is ‘completed’), its completion transitions are evaluated and so a composite transition occurs.

A transition without a target specified is known as an internal transition, when this is traversed the state is not exited or entered and only the traversal effect is performed.

A self-transition can be created by specifying the source and target as the same state, in which case the state is exited, transition effect is performed and then the state is re-entered.

Transitions are not limited to SimpleStates and PseudoStates in the same containing Region or CompositeState; external transitions are allowed that jump across the state machine hierarchy. In these cases, the Exit operation cascades to up to, but not including the least common ancestor in the state machine hierarchy followed by a cascade of the Entry operation to the target state or pseudo state.

Transition Class

An untyped transition from a state or pseudo state to another state or pseudo state tested for upon entry to the source state or pseudo state.

Inheritance Hierarchy

System.Object

Steelbreeze.Behaviour.Transition

Syntax

```
public class Transition
```

The Transition type exposes the following members:

Constructors

Name	Description
Transition(PseudoState, PseudoState, Func<Boolean>)	Initialises a new instance of the Transition class from a PseudoState to a PseudoState with an optional guard condition.
Transition(PseudoState, SimpleState, Func<Boolean>)	Initialises a new instance of the Transition class from a PseudoState to a SimpleState with an optional guard condition.
Transition(SimpleState, PseudoState, Func<Boolean>)	Initialises a new instance of the Transition class from a SimpleState to a PseudoState with an optional guard condition.
Transition(SimpleState, SimpleState, Func<Boolean>)	Initialises a new instance of the Transition class from a SimpleState to a SimpleState with an optional guard condition.

Methods

Name	Description
OnEffect	Protected method allowing sub-classes to override transition behaviour.

Events

Name	Description
Effect	Occurs when a Transition is traversed.

Remarks

A Transition behaves in a similar manner to a Transition<T> however its trigger is the entry to a state or pseudo state rather than a message. Hence, it has no type associated with it.

Upon entry to a PseudoState or SimpleState (or derivative) that is 'completed', the guard conditions of all completion transitions are evaluated and if a single one found, it is traversed. If multiple are found, the machine is considered malformed and an exception is thrown.

If the source is a PseudoState and no completion transitions are found the machine is considered malformed and an exception is thrown.

Transition.Else Class

A catch-all completion transition from junction or completion pseudo states used where no other transitions guards evaluate true.

Inheritance Hierarchy

System.Object

Steelbreeze.Behaviour.Transition

Steelbreeze.Behaviour.Transition.Else

Syntax

```
public class Transition { public sealed class Else : Transition }
```

The Transition.Else type exposes the following members:

Constructors

Name	Description
Transition(PseudoState, PseudoState)	Initialises a new instance of the Transition.Else class from a PseudoState to a PseudoState.
Transition(PseudoState, SimpleState)	Initialises a new instance of the Transition.Else class from a PseudoState to a SimpleState.

Events

Name	Description
Effect	Occurs when a Transition.Else is traversed.

Remarks

A Transition.Else is a special type of completion transition available as a catch-all for circumstances where the guard conditions of regular completion transition may not cater for every eventuality.

It is highly recommended to use Transition.Else completion transitions where you cannot guarantee 100% coverage of guard conditions.

IState Interface

An interface from which to implement a container for the state of the state machine hierarchy.

Syntax

```
public interface IState
```

The IState type exposes the following members:

Methods

Name	Description
SetActive(Element, Boolean)	Updates the active state of an element within a state machine hierarchy.
GetActive(Element)	Returns the active state of an element within a state machine hierarchy.
SetCurrent(Element ,SimpleState)	Updates the child state of a Region or CompositeState within a state machine hierarchy.
GetCurrent(Element)	Returns the child state of a Region or CompositeState within a state machine hierarchy.

Properties

Name	Description
IsTerminated	Boolean indicating that a state machine has been terminated (reached a Terminate PseudoState during execution).

Remarks

The implementation of state.cs has separated a state machine into its model its state. It is side-effect free with the exception of updates to implementations of the IState interface (the state). Therefore, control of transactions, serialisation, threading etc. is entirely up to the developer of the particular state machine.

PseudoStateKind Enumeration

Defines the behaviour of an individual PseudoState.

Syntax

```
public enum PseudoStateKind
```

Membes

Member Name	Description
Choice	Enables a dynamic conditional branches; within a compound transition.
DeepHistory	A type of initial pseudo state; forms the initial starting point when entering a region or composite state for the first time.
Initial	A type of initial pseudo state; forms the initial starting point when entering a region or composite state.
Junction	Enables a static conditional branches; within a compound transition.
ShallowHistory	A type of initial pseudo state; forms the initial starting point when entering a region or composite state for the first time.
Terminate	Entering a terminate PseudoState implies that the execution of this state machine by means of its context object is terminated.

Remarks

The kind of a PseudoState defines its behaviour.

Initial, ShallowHistory and DeepHistory are all ‘initial’ kinds, meaning they define the behaviour when entering a Region or CompositeState. Upon the first entry to the owning Region or CompositeState the initial PseudoState will be entered and its single completion transition traversed. Subsequent entry to the owning Region or Composite state will have the following behaviour:

- Initial: as per the first entry, the Initial Pseudo state will be entered.
- ShallowHistory: the last known child state of the owing Region or CompositeState will be entered.
- DeepHistory: as per ShallowHistory, but all child Regions or CompositeStates will have history semantics applied as well.

Choice and Junction PseudoStates are used to break up transitions to enable complex logic and behaviour; they differ where multiple outbound guard conditions evaluate true:

- Choice will select one of the matching transitions at random.
- Junction will consider the state machine to be malformed and throw an exception.

The use of Choice PseudoStates is ill-advised where repeatable behaviour is required.