YaFSM Basics

by Jörg Kreuzberger

24.03.10

Inhaltsverzeichnis

1 Purpose of this document

The document describes the basic ideas of the Ya Finite State Machine code generator.

2 Refered Documents

1: , ,

3 Abbrevations

FSM	Finite State Machine

4 YaFSM

The purpose of the YaFSM Generator is to generate code for a FSM through an XML based FSM design description. The developers focus should be on the design, not on the coding of the designs implementation.

4.1 FSM Definitions

There are several different FSM implementations and designs available, each one perfectly fitting to special use cases.

The YaFSM generator follows the State Pattern described in [1].

4.1.1 States

The YaFSM generator XML description uses hierachical states. This means any state can have one more more sub-states. If a state (parent state) has substates and the parent state is entered, one of the substates marked as „entry“ state is automatically also entered. If the parent state is exited, the substates are not explicitly exited.

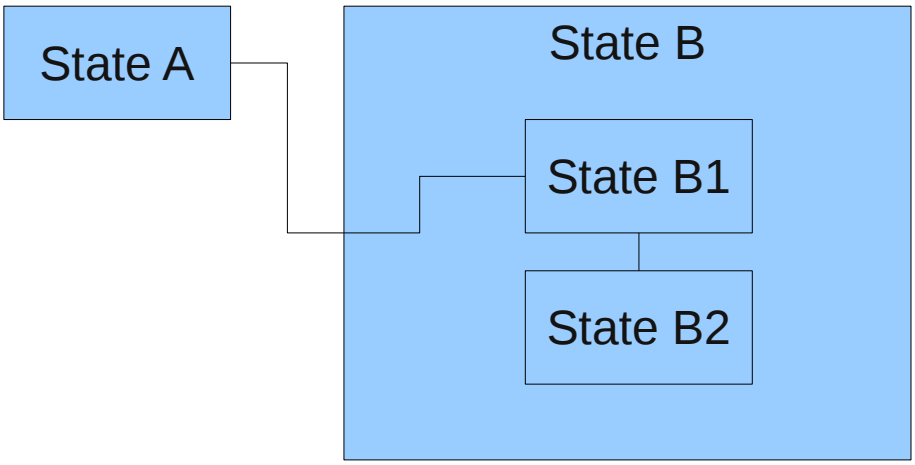

Illustration 1: States with sub-states

4.1.1.1 State Enter

If a state is entered, several activities are started as defined in the XML design description

1. Enter Action: the defined action(s) are called
2. Timer Start: the defined timers are started
3. Timer Stop: the defined timers are stopped

4.1.1.2 State Exit

If a state is exited, several activities are started as defined in the XML design description

1. Exit Action: the defined action(s) are called
2. Timer Start: the defined timers are started
3. Timer Stop: the defined timers are stopped

4.1.2 Triggers

Triggers are method calls to the FSM implementation to trigger the FSM, i.e. to switch states in the FSM.

4.1.2.1 Parameter

Triggers are allowed to have parameters defined in the XML desing description. Due to the generated code language (C++,java, C) it is not possible to define triggers with duplicate names and different parameters, as it would be allowd within the language.

4.1.3 Transitions

Transitions are state changes caused by a trigger, an event, or a timer timeout.

Figure 2 shows an example for a „full-featured“ Ya transition:

The trigger with name „enable“ leads to a transition from state A to state B only if the parameter for the trigger is false. If the trans is executed, the action „OnDisable()“ is called. An Event „updateGui“ is thrown to handle a generic GUI update.

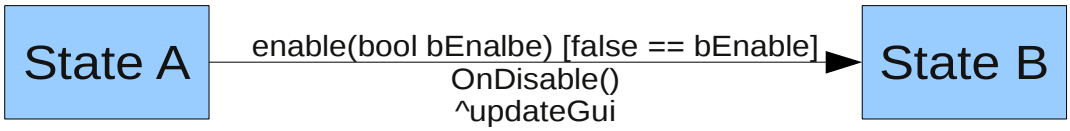

Illustration 2: Example for a FSM transition

4.1.3.1 Self Transitions

A transition must not lead to a state change. A transition may also be defined from state A to state A. This „Self Transitions“ behave like normal transitions regarding actions and events, but do not lead to a state exit and enter call like an transition between different states

4.1.3.2 Conditions

Transition defintions are allowed to have conditions, so the same trigger could be used for different transitions depending on the parameters

4.1.3.3 Actions

A transition can have defined actions, that should be performed on execution of the transition

4.1.3.4 Events

A transition could start an asynchronous event

4.2 State transition

4.2.1 On same hierachy level

On execution of a transition between different states, the defined activities are started in following order:

1. exit action State A
2. exit start timers State A
3. exit stop timers State A
4. transition action
5. transition event fire
6. enter action State B
7. enter start timers State B
8. enter stop timers State B
9. event handling

4.2.2 Enter a state with Substates

1. exit action State A
2. exit start timers State A
3. exit stop timers State A
4. transition action
5. transition event fire
6. enter action State B
7. enter start timers State B
8. enter stop timers State B
9. enter action SubState B1
10. enter start timers SubState B1
11. enter stop timers SubState B1
12. event handling

4.2.3 Exit a state with Substates

1. exit action State B

2. exit start timers State B
3. exit stop timers State B
4. transition action
5. transition event fire
6. enter action State A
7. enter start timers State A
8. enter stop timers State A

Be aware that no exit action of the substates is called any time the parent state is exited.