TUTORIAL 2

Tutorial para a construção de árvores de comportamento com a biblioteca py-trees

Raiz

O nó raiz é primeiro nó da árvore de comportamento. Um exemplo da construção de uma árvore de comportamento é apresentado a seguir.

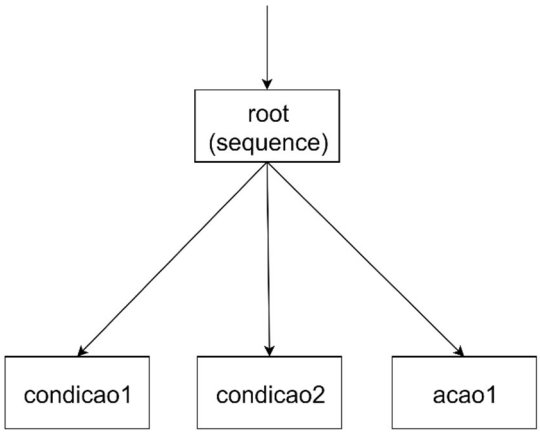
root = py_trees.composites.Sequence(name="Sequence", memory=True)

root.add_child(condicao1)

root.add_child(condicao2)

root.add_child(acao1)

Nesse exemplo, o nó raiz é da classe Sequence.



Sequence

Um nó de controle (objeto) da classe Sequence (não reativo) é criado do seguinte modo

sequence1 = py_trees.composites.Sequence(name="Sequence", memory=True)

O número 1 foi usado no nome do objeto para diferenciá-lo de outros objetos da mesma classe, que podem ser usados em outras partes da árvore de comportamento. Por exemplo, nós podemos criar um outro objeto da classe Sequence, utilizando o número 2.

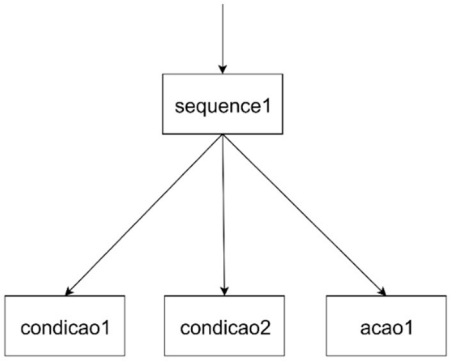
sequence2 = py_trees.composites.Sequence(name="Sequence", memory=True)

Vamos supor que há três objetos: nó condicao1; nó condicao2; e nó acao1. Nós podemos adicioná-los como filhos do nó sequence1, por exemplo, do seguinte modo

sequence1.add_child(condicao1)

sequence1.add_child(condicao2)

sequence1.add_child(acao1)



Isto significa que o nó acao1 será acionado (ticado) apenas se os nós condicao1 e condicao2 retornarem SUCCESS.

Fallback (Selector)

Um nó de controle (objeto) da classe Selector (não reativo) é criado do seguinte modo

fallback1 = py_trees.composites.Selector(name="Selector", memory=True)

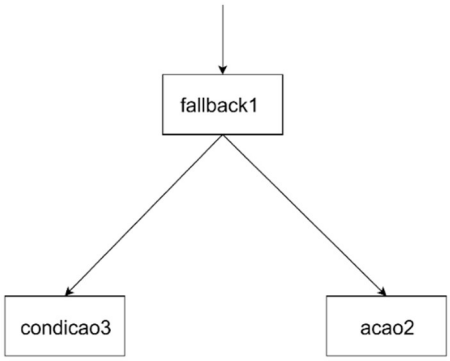
De modo similar aos exemplos descritos para a classe Sequence, o número 1 foi usado no nome do objeto para diferenciá-lo de outros objetos da mesma classe, que podem ser usados em outras partes da árvore de comportamento. Por exemplo, nós podemos criar um outro objeto da classe Selector, utilizando o número 2.

fallback2 = py_trees.composites.Selector(name="Selector", memory=True)

Vamos supor que há dois objetos: nó condicao3; e nó acao2. Nós podemos adicioná-los como filhos do nó fallback1, por exemplo, do seguinte modo

fallback1.add_child(condicao3)

fallback1.add_child(acao2)



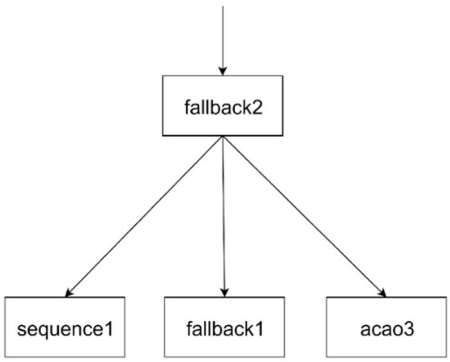
Isto implica que o nó acao2 será acionado (ticado) apenas se o nó condicao3 retornar FAILURE.

Nós podemos adicionar nós de controle como filhos de um outro nó de controle. Por exemplo, podemos adicionar o nó sequence1, o nó fallback1, e um nó acao3 como filhos do nó fallback2.

fallback2.add_child(sequence1)

fallback2.add_child(fallback1)

fallback2.add_child(acao3)



Reactive Sequence

Nós de controle (objetos) da classe Reactive Sequence (reativo) são criados do seguinte modo

reactive_sequence1 = py_trees.composites.Sequence(name="Sequence", memory=False)

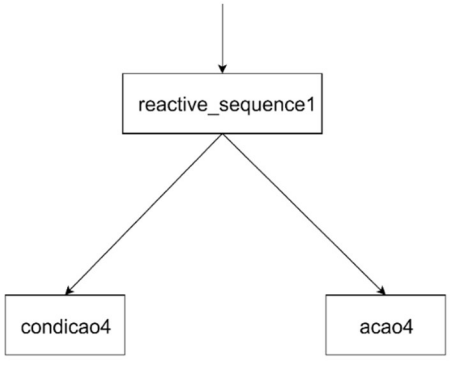
reactive_sequence2 = py_trees.composites.Sequence(name="Sequence", memory=False)

A diferença para o Sequence não reativo é que o reativo não tem memória do nó-filho que retorna RUNNING. Assim, o nó de controle da classe Reactive Sequence reinicia do nó-filho mais à esquerda quando um nó-filho retorna RUNNING no tick anterior.

Vamos supor que há dois objetos: nó condicao4; e nó acao4. Nós podemos adicioná-los como filhos do nó reactive_sequence1, por exemplo, do seguinte modo

reactive_sequence1.add_child(condicao4)

reactive_sequence1.add_child(acao4)



Reactive Fallback (Reactive Selector)

Nós de controle (objetos) da classe Reactive Fallback (reativo) são criados do seguinte modo

reactive_fallback1 = py_trees.composites.Selector(name="Selector", memory=False)

reactive_fallback2 = py_trees.composites.Selector(name="Selector", memory=False)

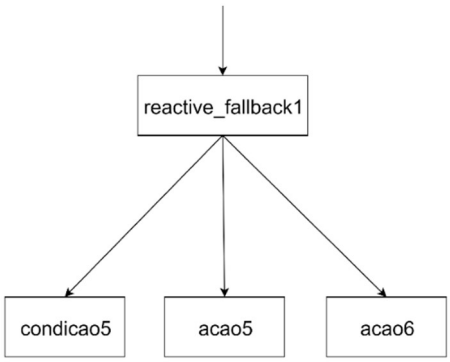
A diferença para o Fallback não reativo é que o reativo não tem memória do nó-filho que retorna RUNNING. Assim, o nó de controle da classe Reactive Fallback reinicia do nó-filho mais à esquerda quando um nó-filho retorna RUNNING no tick anterior.

Vamos supor que há três objetos: nó condicao5; nó acao5; e nó ação 6. Nós podemos adicioná-los como filhos do nó reactive_fallback1, por exemplo, do seguinte modo

reactive_fallback1.add_child(condicao5)

reactive_fallback1.add_child(acao5)

reactive_fallback1.add_child(acao6)



Inverter

Um nó de controle (objeto) da classe Inverter é criado do seguinte modo

name="Inverter"

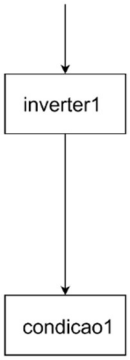
inverter_name = py_trees.decorators.Inverter(name, name_child)

O nome do nó-filho é o nome de um nó criado anteriormente. Um exemplo é apresentado a seguir.

name="Inverter"

inverter1 = py_trees.decorators.Inverter(name, condicao1)

Considerando que condicao1 é um objeto criado anteriormente.



Exemplo de uma Árvore de Comportamento

root = py_trees.composites.Sequence(name="Sequence", memory=True)

root.add_child(condicao1)

fallback1 = py_trees.composites.Selector(name="Selector", memory=True)

fallback1.add_child(condicao2)

fallback1.add_child(condicao3)

root.add_child(fallback1)

Nesse exemplo, o nó raiz é da classe Sequence.