CT-213
Laboratório 01- Máquina de Estados Finita e Behavior Tree
(11 de março de 2020)

André Moreno[1]
[1]Aluno de Graduação do Instituto Tecnológico de Aeronáutica, São José dos Campos, Brasil.
E-mail: moreno6230@gmail.com

1 Máquina de Estados Finita

A fim de obter os resultados propostos, no construtor de cada classe criou-se uma variável tempo e a inicializou com o valor zero. Ademais, realizou-se a implementações descritas a seguir

1.1 MoveForwardState

No método *check_transition* foram realizadas as seguintes implementações:

- Se o método *get_bumper_state* retornasse *True*, o estado mudaria para o estado *GoBackState*;
- Se o *Roomba* permanecesse nesse estado por um tempo superior a *MOVE_FORWARD_TIME* (um intervalo de tempo pré-determinado), mudaria para o estado *MoveInsSpiralState*.
- Por fim, a variável tempo era incrementada pelo constante fornecida *SAMPLE_TIME*.

No método *execute*:

- utilizou-se *set_velocity* para atribuir uma velocidade linear ao *Roomba* para ele percorresse em linha reta.

1.2 MoveInSpiralState

O método *check_transition* desta classe foi implementado de forma análoga ao *MoveForwardState*:

- Se o *get_bumper_state* retornasse *True*, o estado mudaria para o estado *GoBackState*;
- Se o *Roomba* permanecesse nesse estado por um tempo superior a *MOVE_IN_SPIRAL_TIME*, ele retorna para o estado *MoveForwardState*.
- Por fim, a variável tempo era incrementada pelo constante fornecida *SAMPLE_TIME*.

No método *execute* desta classe:

- utilizou-se o método *set_velocity* para atribuir um movimento em espiral, utilizando como parâmetro para velocidade linear *FORWARD_SPEED* e para velocidade angular a seguinte relação:

$$a = FORWARD\_SPEED$$

$$b = INITIAL\_RAIDUS\_SPIRAL$$

$$c = SPIRAL\_FACTOR$$

$$VELOCIDADEANGULAR = \frac{a}{b + c * tempo}$$

1.3 GoBackState

Neste estado, no método *check_transition* foram realizadas as seguintes implementações:

- se o tempo que o objeto estava neste estado era superior a *GO_BACK_TIME*, o estado mudaria para o estado *RotateState*.
- Por fim, incrementava-se a variável tempo.

No método *execute* desta classe, por sua vez:

- atribuiu-se uma velocidade linear negativa e velocidade angular nula, fazendo com que o Roomba retornasse uma pequena distância, utilizando o método *set_velocity*.

1.4 RotateState

No construtor desta classe, além de inicializar uma variável utilizada para controla o tempo, atribuindo o valor zero, utiliza a função random para obter um número aleatório entre $(\frac{-\pi}{ANGULAR\_SPEED})$ e $(\frac{\pi}{ANGULAR\_SPEED})$ que será utilizado como o limite de tempo que o *Roomba* ficará rotacionando em torno do seu eixo até mudar para o próximo estado.

No método *check_transition*:

- Caso o tempo neste estado for superior ao tempo obtido anteriormente, o estado mudaria para o estado *MoveForwardState*.
- Após, incrementava-se a variável tempo.

A função *execute*, por sua vez:

- atribui, por meio da função *set_velocity*, uma velocidade linear nula e uma velocidade angular igual a *ANGULAR_SPEED*.

1.5 Imagens do funcionamento

Observa-se nas Fig. 1, 2 e 3 o movimento do *Roomba* e a mudança de seu comportamento de acordo com a mudança de estado.

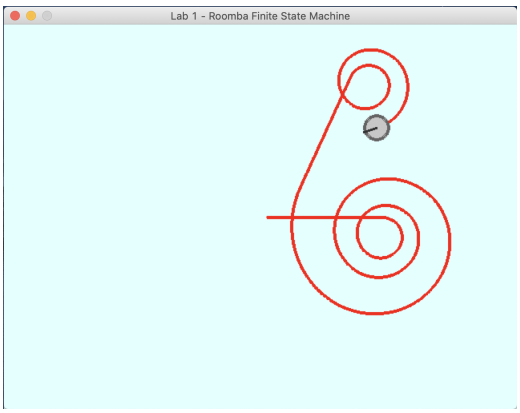

Figura 1: State Machine - Estado MoveForwardState e Estado MoveInSpiralState

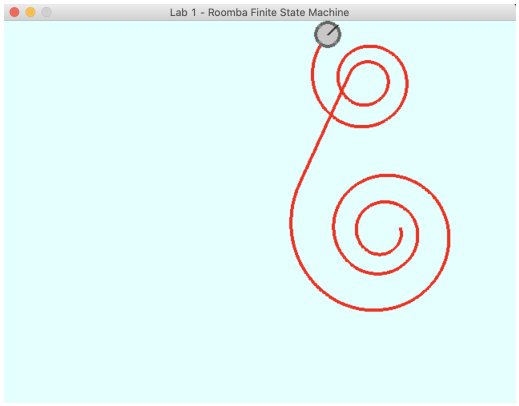

Figura 2: State Machine - Estado GoBackState após a colisão

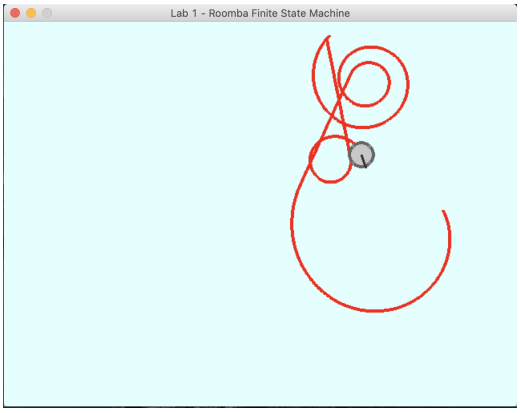

Figura 3: State Machine - Estado RotateState - Estado MoveForwardState

2 Behavior Tree

Para a implementação da Behavio Tree, fez-se necessário, inicialmente, a construção da árvore de comportamento. Dessa forma, utilizando o método *SequenceNode*, criou-se os nós *root*, *node1* e *node2*.

Após, utilizou-se o método *add_child*, para adicionar os nós *node1* e *node2* como filhos de *root*, para adicionar os nós *MoveForwardNode* e *MoveInSpiralNode* como filhos de *node1* e os nós *GoBackNode* e *RotateNode* como filhos de *node2*.

Após a construção da árvore de comportamento, implementou-se cada folha da árvore conforme descrito nos tópicos abaixo.

2.1 MoveForwardNode

2.1.1 Enter

Neste tópico, apenas inicializou uma variável com função de tempo com o valor zero. e atribuiu-se ao *Roomba* uma velocidade linear como o valor proposto e uma velocidade angular nula utilizando o método *set_velocity*.

2.1.2 Execute

Inicialmente verificou-se se a função *get_bumper_state* é verdadeira. Caso sim, o método retorna *FAILURE*, e assim, a função o direcionará para o nó *GoBackNode*.

Caso o tempo que o robô execute a função for superior ao esperado, a função retornará *SUCCESS* ele irá para o nó *MoveInSpiralNode*.

Caso não tenha ocorrido nenhuma dessas duas situações, incrementava-se a variável tempo e o método retorna *RUNNING*.

2.2 MoveInSpiralNode

2.2.1 Enter

Neste método apenas inicializou uma variável com o valor zero para controlar o tempo que o robô irá permanecer neste nó.

2.2.2 Execute

Inicialmente verificou-se se a função *get_bumper_state* é verdadeira. Caso sim, o método retorna *FAILURE*, e assim, a função o direcionará para o nó *GoBackNode*.

Caso o tempo que o robô execute a função for superior ao esperado, a função retornará *SUCCESS* ele irá para o nó *MoveForwardNode*.

Caso não tenha ocorrido nenhuma dessas duas situações, significa que o robô deve continuar no mesmo percurso, dessa forma, atribui uma velocidade por meio da função *set_velocity* de forma análoga a explicado no MoveInSpiralState na seção 1.2 e incrementa a função tempo. A função, então, retorna *RUNNING*.

2.3 GoBackNode

2.3.1 Enter

Análogo ao nó *MoveForwardNode*, diferenciando-se que foi atribuída uma velocidade linear negativa ao robô.

2.3.2 Execute

Inicialmente, verifica se o tempo que o robô esta no nó é superior ao tempo pré-determinado. Caso sim, o método retornará *SUCCESS* e irá para o nó *RotateNode*.

Caso contrário, ele continuará no percurso programado pelo nó, a variável tempo será incrementada e o método retornará *RUNNING*.

2.4 RotateNode

2.4.1 Enter

Além de inicializar a variável tempo com valor zero, analogamente na seção 1.4, obteve-se aleatoriamente um valor que será utilizado para determinar o tempo que o robô ira girar em torno do seu eixo.

2.4.2 Execute

Inicialmente, verifica se o tempo que o robô esta no nó é superior ao tempo pré-determinado.

Caso sim, o método retornará *SUCCESS* e irá voltar para o nó *MoveForwardNode*.

Caso contrário, ele continuará no rotacionando e irá incrementar a variável tempo. A função, então, retorna *RUNNING*.

2.5 Imagens do Funcionamento

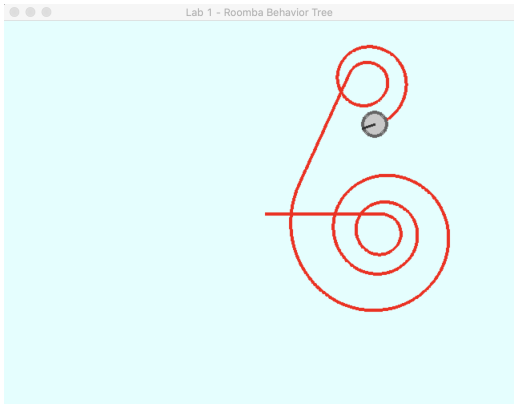

Figura 4: Behavior Tree - MoveForwardNode e MoveInSpiralNode

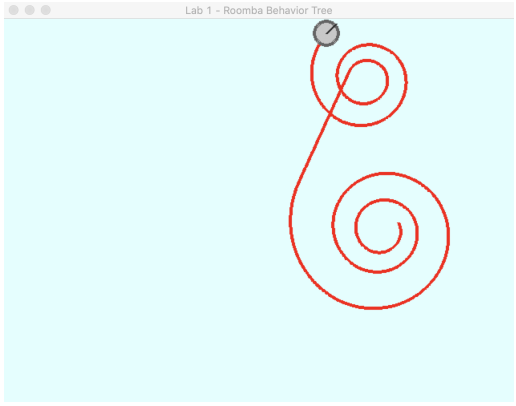

Figura 5: Behavior Tree - GobackNode

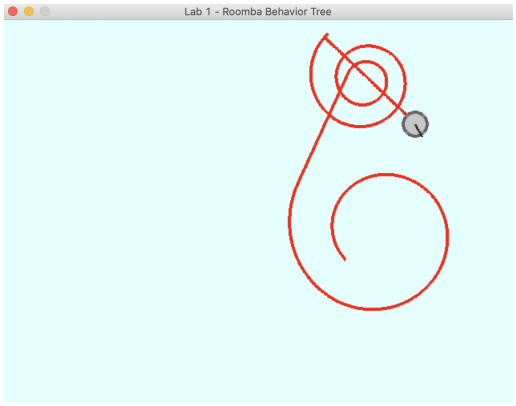

Figura 6: Behavior Tree - RotateNode e MoveForwardNode