Instituto Tecnológico de Aeronáutica

CT-213

Laboratório 1 – Máquina de Estados Finita e Behavior Tree

Aluno:
Pedro Elardenberg Sousa e Souza

Professor:
Marcos Ricardo Omena de Albuquerque Maximo

26 de março de 2021

São José dos Campos, Brasil

1 Funcionamento da máquina de estados

Para a máquina de estados, foram implementadas as classes *MoveForwardState, MoveInSpiralState, GoBackState e RotateState*. No *__init__* de cada estado, foram criadas as variáveis contador "cont"e tempo "t", para controlar o tempo em que o robô ficaria naquele estado. Para cada uma delas, a implementação dos métodos *check_transition e execute* foi feita conforme a seguir:

MoveForwardState: o robô mudará para *MoveInSpiralState* caso a variável "t"esteja maior que o MOVE_FORWARD_TIME e para *GoBackState* caso o robô atinja o bumper. No execute, a variável "t"recebe um incremento de valor SAMPLE_TIME e a velocidade do robô é definida na função *set_velocity* com os parâmetros FORWARD_SPEED para velocidade linear e 0 para velocidade angular.

MoveInSpiralState: o robô mudará para *MoveForwardState* caso a variável "t"esteja maior que o MOVE_IN_SPIRAL_TIME e para *GoBackState* caso o robô atinja o bumper. Após contar o tempo conforme a função anterior, o raio da espiral é calculado conforme o raio inicial (constante fornecida) e SPIRAL_FACTOR $\cdot$ "t"e a velocidade angular "angular_speed"pela razão FORWARD_SPEED pelo raio da espiral. A velocidade do robô é dada pela FORWARD_SPEED para velocidade linear e "angular_speed"para velocidade angular.

GoBackState: o robô mudará para o estado *RotateState* caso "t"seja maior que o GO_BACK_TIME. No execute, o robô andará com BACKWARD_SPEED de velocidade linear e 0 de velocidade angular.

RotateState: para este caso, foi gerado um número aleatório "random". Como, no python, essa função gera um número aleatório entre 0 e 1, o ângulo de rotação "rotate_angle"foi calculado a partir de "random"para ser inserido no intervalo $[-\pi, \pi]$. A transição é feita para *MoveForwardState* após que o tempo para o robô fazer a rotação seja cumprido, ou seja, "t$\gg$ "rotate_angle"/ ANGULAR_SPEED. Por fim, no execute, o agente foi definido com uma velocidade linear de 0 e velocidade angular igual a "ANGULAR_SPEED", conforme a orientação do ângulo de rotação.

As imagens abaixo fornecem o funcionamento do *Roomba*, na implementação por máquina de estados.

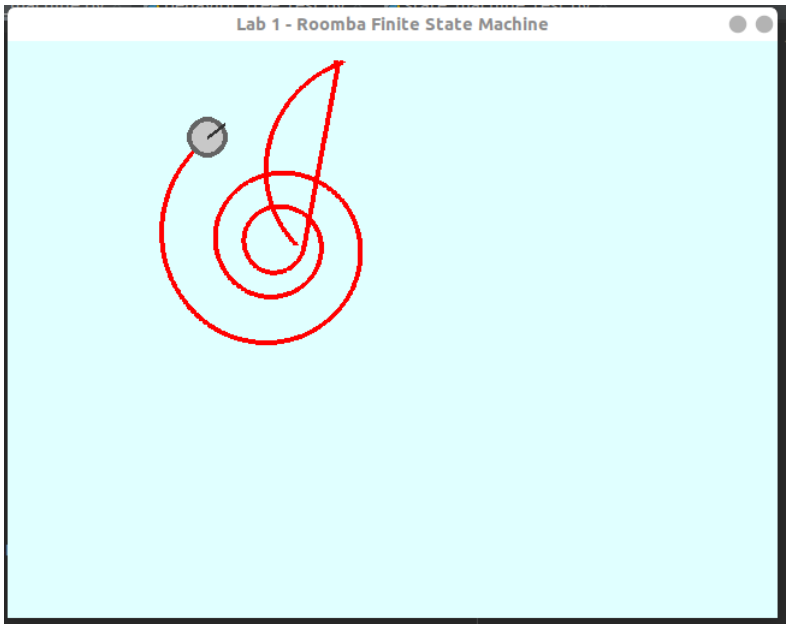

Figura 1: Colisão - MRU - Espiral

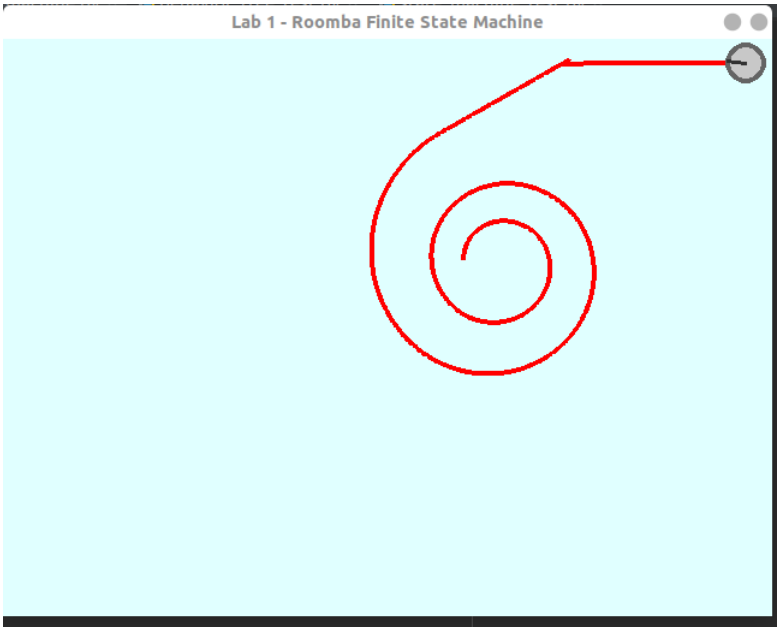

Figura 2: MRU - Colisão - MRU - Colisão

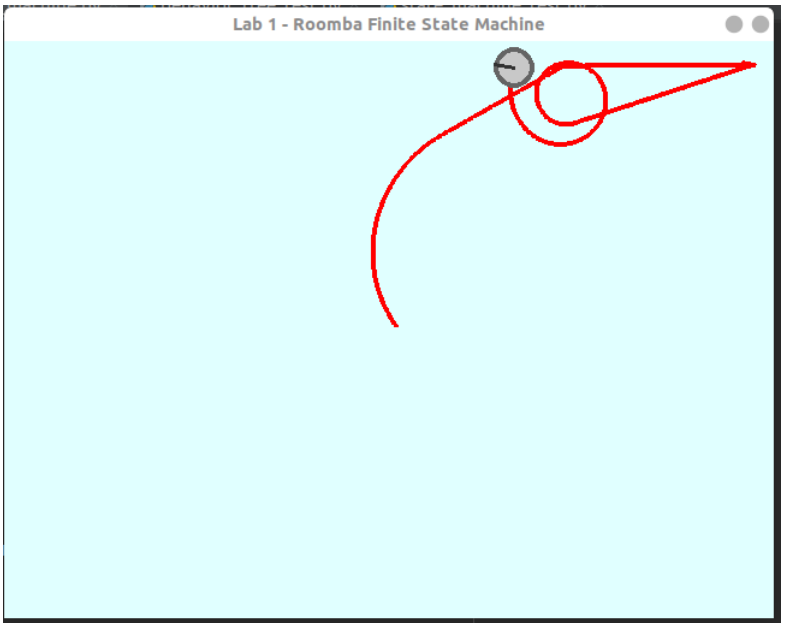

Figura 3: MRU - Espiral - Colisão

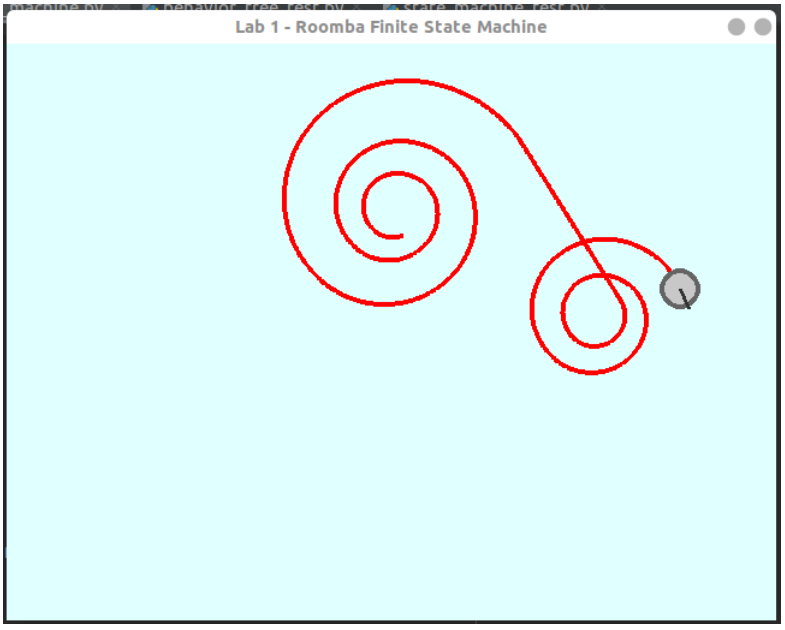

Figura 4: Espiral - MRU - Espiral

2 Funcionamento da behavior tree

Para a behavior tree foram implementadas as classes *MoveForwardNode, MoveInSpiralNode, GoBackNode e RotateNode*. Além disso, na classe *BehaviorTree*, foram criados os nós e os vértices da árvore de comportamento. Novamente, no *__init__* de cada nó foram criados um contador e uma variável "t"para marcarem o tempo que o agente fica em cada nó. Para cada uma delas, a implementação dos métodos enter e execute foi feita conforme a seguir:

MoveForwardNode: quando o agente entra nesse nó, zera-se o contador. No execute, retorna FAILURE no ExecutionStatus caso atinja o bumper, retorna RUNNING caso o tempo seja menor que o MOVE_FORWARD_TIME (no qual define a velocidade do robô conforme no caso anterior) e retorna SUCCESS em último caso.

MoveInSpiralNode: quando o agente entra nesse nó, zera-se o contador. No execute, retorna FAILURE no ExecutionStatus caso atinja o bumper, retorna SUCCESS caso "t"seja maior que MOVE_IN_SPIRAL_TIME e retorna RUNNING, executando a rotina de mover-se em espiral, em útimo caso.

GoBackNode: quando o agente entra nesse nó, zera-se o contador. No execute, retorna RUNNING caso "t"seja menor que o GO_BACK_TIME, executando a rotina de mover-se para trás, e retorna SUCCESS, caso contrário.

RotateNode: quando o agente entra nesse nó, zera-se o contador, além de gerar um ângulo de rotação aleatório, conforme descrito anteriormente. No execute, retorna RUNNING caso "t"seja menor que o tempo de rotação do *Roomba*, executando a devida rotina de rotação pura e retorna SUCCESS, caso contrário.

As imagens abaixo fornecem o funcionamento do *Roomba*, na implementação por behavior tree.

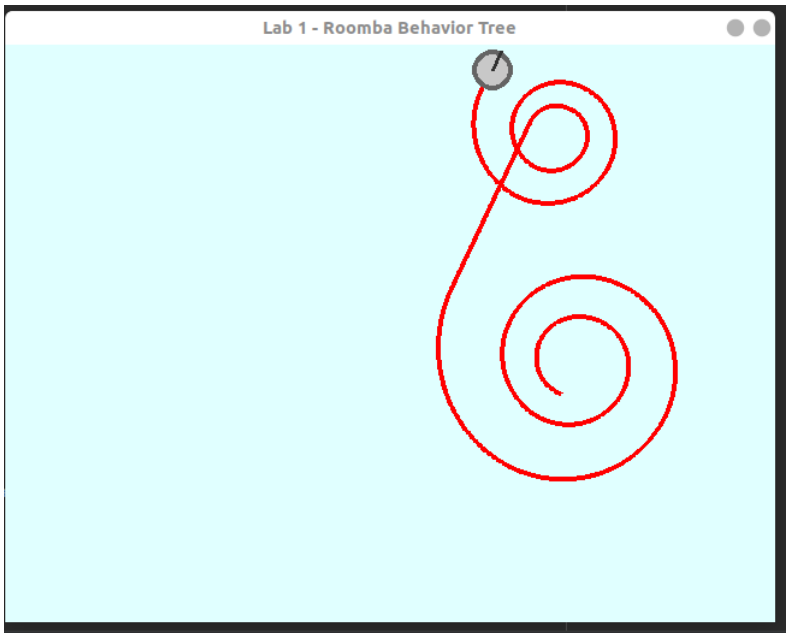

Figura 5: Espiral - MRU - Espiral

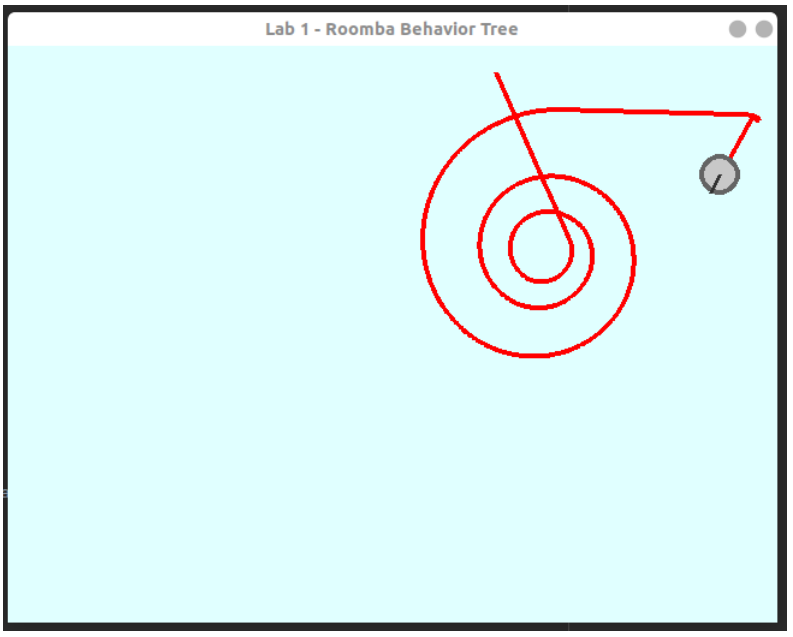

Figura 6: MRU - Espiral - MRU - Colisão

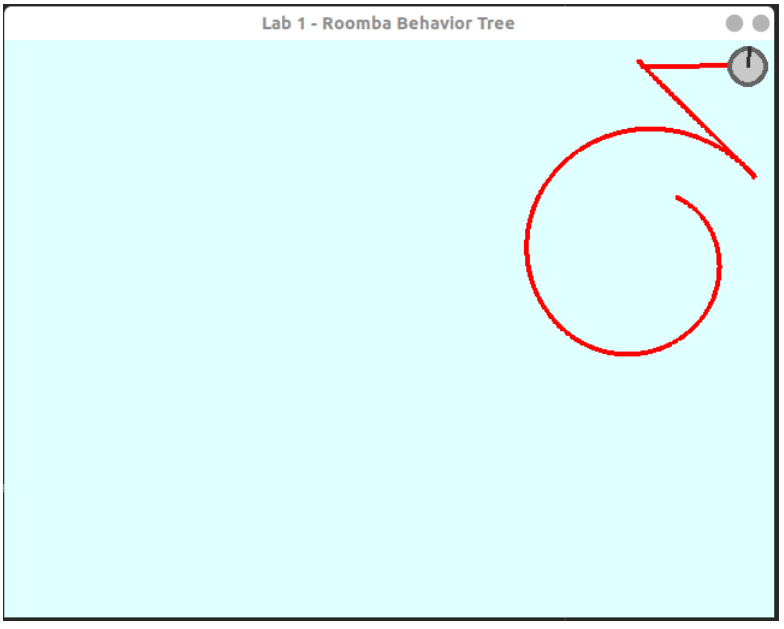

Figura 7: Espiral - Colisão - MRU - Colisão - MRU - Colisão

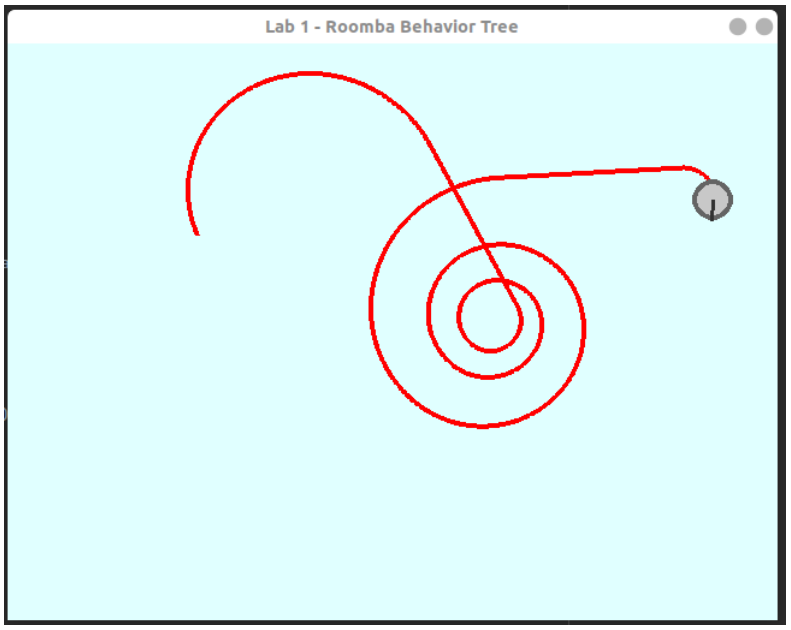

Figura 8: Espiral - MRU - Espiral - MRU