universidade de aveiro

Desenvolvimento de um Agente autónomo para o jogo

Bomberman

Introdução à Inteligência Artificial

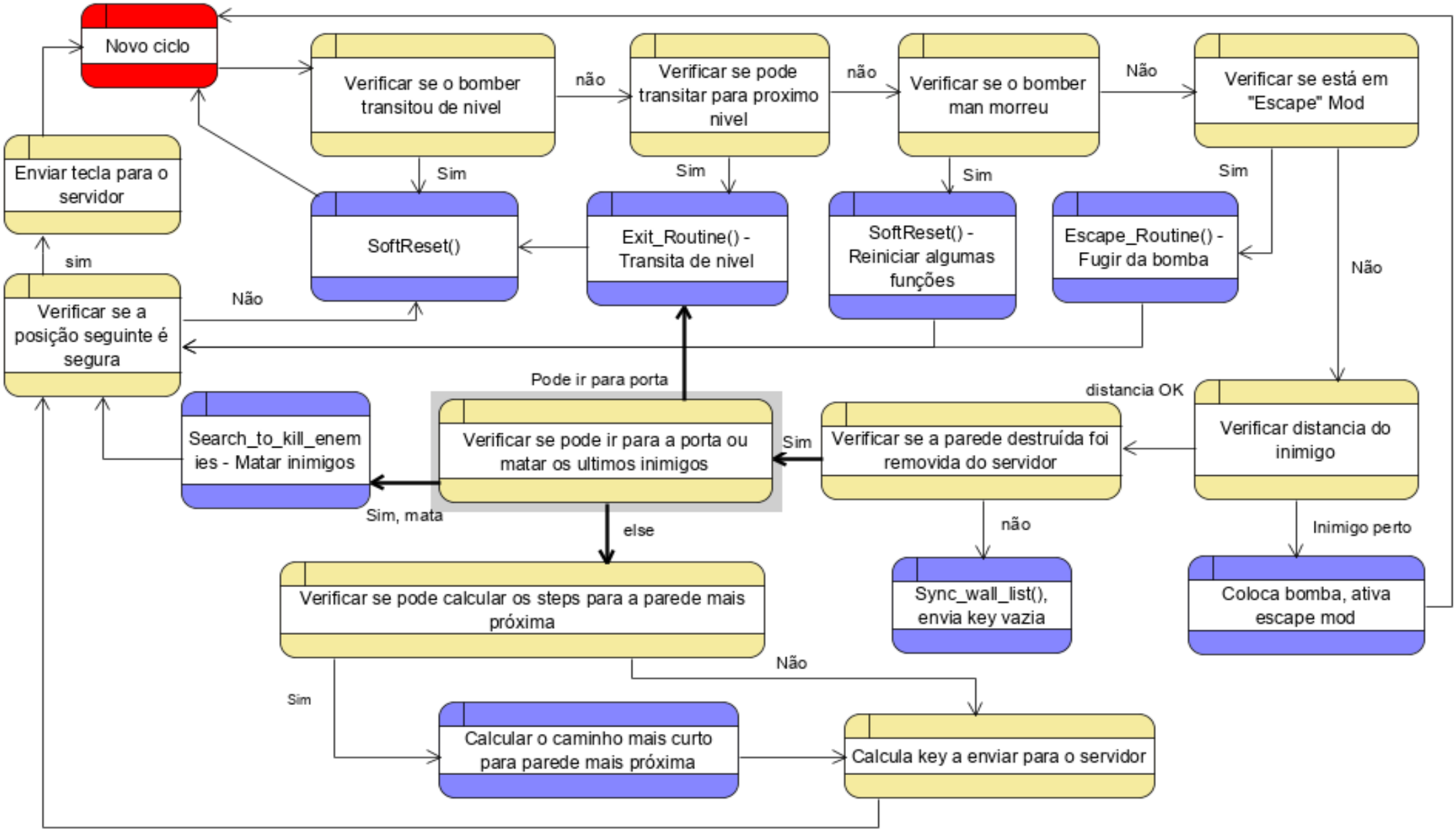
Ano Letivo de 2019/2020

Rodrigo Santos Nº89180
Daniel Lopes Nº87881
Alexey Kononov Nº89227

Repositório: **trabalho-de-grupo-bomberman-bomberman-iia-89180-87881-89227**

4 de dezembro 2019

A estratégia utilizada pelo nosso agente consiste em destruir todas as paredes, tentando matar os inimigos que aparecem pelo caminho do Bomberman (BM), de seguida matar os restantes, apanhar o powerup (dependendo do nível) e prosseguir para a porta. O diagrama seguinta, visa mostrar o funcionamento base do nosso algoritmo



A movimentação do nosso BM é feita através do cálculo do caminho mais curto para a **posição-objetivo** que se encontra armazenada na variável "goal".

As funções que calculam "goal":

1. **calculate_position()**: Calcula as coordenadas da posição ao lado da próxima parede a ser destruída para colocar a bomba	2. **exit_routine()**: caso BM não tenha o powerup -> goal = posição do powerup, caso contrario -> goal = posição da porta	3. Funções para matar os inimigos: **killMinvo()** **KillOneal()** **killBallom()**

Estratégias de exterminação dos inimigos:

Dependendo do inimigo, temos 3 estratégias diferentes:

1. Para os inimigos **Ballom** e **Doll,** o BM é enviado para os cantos do mapa percorrendo-o em sentido do relógio. Quando chegar a um canto, é enviado para o outro e assim sucessivamente. Como o mapa sem paredes destruíveis, é previsível a trajetória dos inimigos.

2. Para os inimigos **Oneal, Kondoria e Ovapi,** a estratégia é igual à anterior, mas desta vez em sentido contrario ao relógio

3. Para os inimigos **Minvo e Pass,** o BM é enviado para a posição dos mesmos, quando a uma distancia próxima deles, o BM coloca a bomba.

A **pesquisa** do caminho mais curto é realizada recorrendo à pesquisa em árvore binária, semelhante à usada nas aulas práticas, usando a estratégia "greedy". Para isso existem 2 ficheiros: **Bomberman.py** **e** **tree_search.py**

Flags (Variáveis globais)

- A nossa implementação em grande parte foca-se no uso de variáveis globais, à qual designamos por flags. Cada função terá o seu comportamento dependendo dos valores de cada flag.

- ***Por exemplo***: Caso o BM esteja a escapar da bomba, existe uma flag "bomb_flag" que estará a valor 1, estando esta flag a 1, a função, por exemplo, de calcular o caminho mais curto, não é executada, assim como outras.

- Para mais detalhes, todas as variáveis globais, estão comentadas no código.

- No slide seguinte, decidimos mostrar mais um esquema, desta vez um diagrama de estados que representa a função escape_routine() ou seja o comportamento do BM quando este coloca a bomba e escapa. O BM quando coloca a bomba procura sempre o **canto** mais próximo a se esconder (prioridade principal) caso não consiga, opta por outra outra solução (prioridade secundária).

Diagrama de estados - Escape_routine