Universidade Estadual do Sudoeste da Bahia
Bacharelado em Ciência da Computação

Super Mario Wumpus – Tutorial

Trabalho complementar da disciplina Sistemas Inteligentes.

O Super Mario Wumpus é uma interface de manipulação e testes feita em JAVA, que possui como objetivo simular o resultado de soluções inteligentes para resolver o problema computacional: mundo wumpus. Esta foi desenvolvida com base em um clássico dos videogames, o super mario bros.
Para utilizar o recurso o usuário deve importar a interface para a sua classe e controlá-la através de métodos públicos disponíveis no objeto. Outra forma de utilização é a manipulação manual ou direta que pode ser feita configurando a interface durante seu estado inicial.
O objeto principal da interface é a classe Game.java que pode ser criado através do código *Game game = new Game( );* .
Para iniciar a interface é necessário utilizar um método especial chamado init( ), que pode ser facilmente acessado acessando o objeto game criado. O código de inicialização então é *game.init( );* .
Ao atingir este ponto do código a interface será iniciada e mostrará uma tela inicial de apresentação e configurações como mostra a *figura 1*.

figura 1

Na tela inicial quatro opções são apresentadas: 1 começar jogo, 2 opções, 3 sobre e 4 sair. Todas essa opções com exceção da ultima, que encerra a interface, levam o usuário a uma nova tela.
O botão “começar jogo” dispara o processo de simulação do problema do mundo wumpus iniciando uma nova tela que apresenta o ambiente de interação. Como mostra a *figura 2* a tela de jogo permite ao algorítimo ou controle manual do usuário controlar o personagem que viaja pelo mapa em busca do seu objetivo.

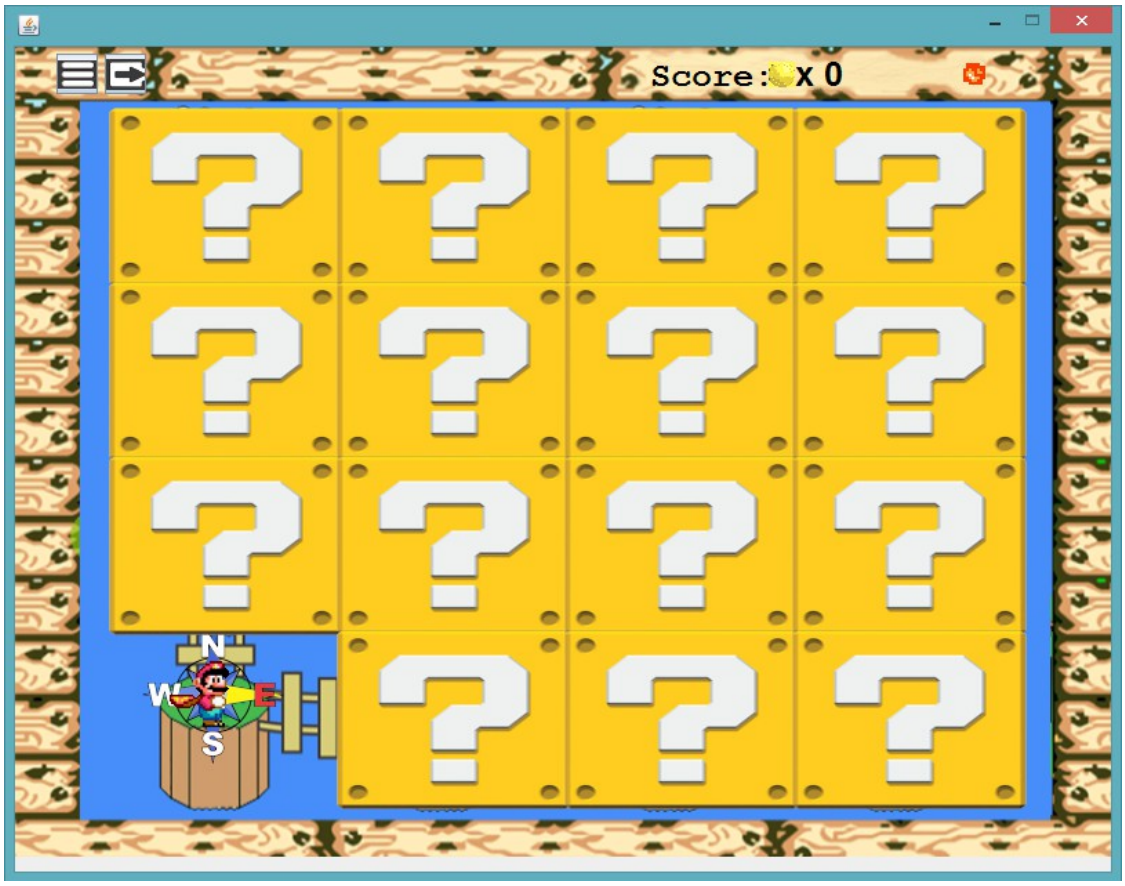

Figura 2

A pontuação do jogador, assim como a munição, pode ser observada no topo da tela.
O botão “opções” permite ao usuário fazer algumas alterações visuais na interface. A tela apresentada na *figura 3* possibilita a edição da velocidade de animação, da quantidade de poços existentes no mapa do jogo e dos efeitos de áudio disponíveis.
Já o botão “sobre” passa o usuario para uma tela que oferece algumas informações sobre o problema do mundo wumpus e sobre os autores da interface.

>>Controles Manuais

Para fazer o controle manual do personagem no ambiente o usuário deve habilitar esta opção na tela de opções.
O controle das funções do personagem são dados através do acionamento de teclas específicas na tela de jogo. Veja abaixo uma relação entre as teclas utilizadas e suas respectivas ações.

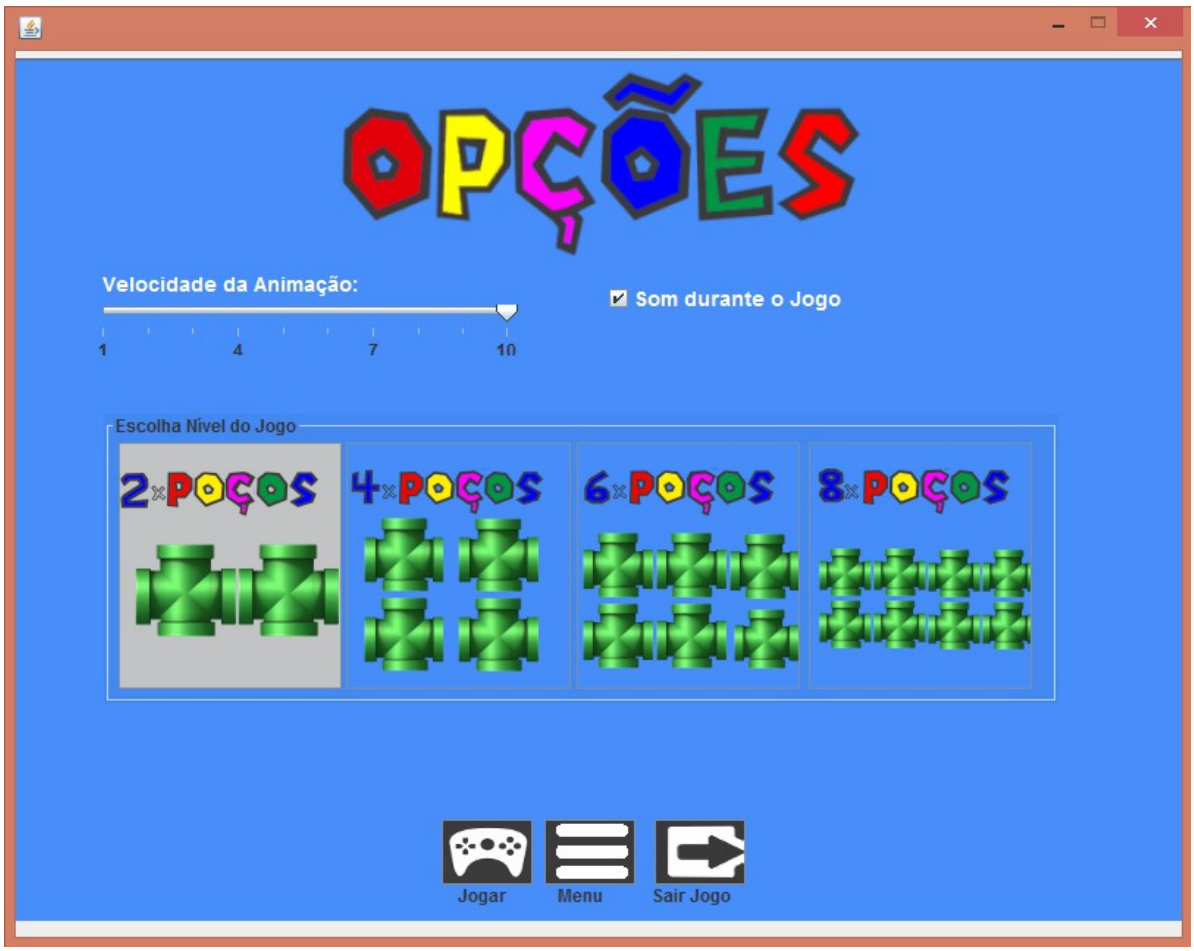

figura 3

enter – faz o personagem andar para a direção indicada em seu menu de direções.
arrow_left – faz o personagem mudar sua direção para a próxima à esquerda.
arrow_right – faz o personagem mudar sua direção para a próxima à direita.
space – pega um objeto no ambiente.
Ctrl – atira uma bola de fogo na direção indicada no menu do personagem.

Observação: O personagem desbloqueia uma nova posição do mapa assim que a alcança.

>>Controle por código

Para controlar o personagem por código o programador deve utilizar alguns métodos disponíveis na interface. Confira abaixo a lista de métodos e suas funções:
1 – guerreiroAndar();
método do tipo void que permite ao personagem andar de uma sala para outra seguindo uma direção predefinida.
2 – guerreiroGirarSentHorario();
método do tipo void que possibilita ao personagem girar no sentido horário alcançando uma nova direção de destino.
3 – guerreiroGirarSentAntiHorario();

método do tipo void que possibilita ao personagem girar no sentido anti-horário alcançando uma nova direção de destino.
4 – guerreiroAtirar();
método tipo booleano que permite ao personagem disparar um tiro de fogo em uma direção predefinida. O retorno verdadeiro do método indica que o tiro acertou o monstro e o retorno falso indica que não o acertou.
5 – guerreiroPegarObjeto();
método tipo booleano que possibilita ao personagem buscar um objeto na sala atual. O retorno verdadeiro indica que o objeto foi encontrado e capturado e o retorno falso indica que nada foi encontrado.
6 – isFedor();
método tipo booleano que aponta se a sala atual possui fedor. Retorno verdadeiro indica a presença do fedor e o falso que não há fedor.
7 – isBrisa();
método tipo booleano que diz se existe brisa na sala atual. Retorno verdadeiro indica que tem brisa e o falso que não tem.
8 – isResplendor();
método tipo booleano que indica se existe brilho ou resplendor na sala atual indicando assim a presença do objeto(do ouro).O retorno verdadeiro indica que tem resplendor e o falso indica que não tem.
9 – isFlecha();
método tipo booleano que permite ao usuário saber se o guerreiro possui munição para atirar contra o monstro. Caso verdadeiro quer dizer que tem munição, caso falso quer dizer que não tem.

Observação: o sensor de impacto impede que o personagem ande para um ponto além da borda do mapa do ambiente. O sensor de som indica através do método guerreiroAtirar() se escutou ou não o grito do wumpus.