

Descrição do 2º trabalho da disciplina

Detalhes do trabalho


O objetivo deste trabalho é implementar um jogo baseado em labirintos. O jogo implementado neste trabalho é um jogo *single-player*, onde o usuário será representado por um avatar, o qual será inserido no labirinto e deverá encontrar a saída. Além do player, inimigos devem ser inseridos em algumas posições do labirinto. O objetivo de um inimigo é encontrar o jogador e impedir que o mesmo chegue até saída.

É importante ressaltar que tanto o player como os inimigos devem se mover apenas por caminhos válidos (não podem atravessar paredes). Portanto, medidas para evitar acesso indevido a posições bloqueadas devem ser implementadas.

Elementos do jogo

- **Player:** possui uma posição inicial no labirinto. O player poderá se mover de acordo com a entrada de dados fornecida via teclado (*w*: cima, *s*: baixo, *a*: esquerda, *d*: direita). Se o player chegar a saída do labirinto o jogo termina.
- **Inimigos:** devem ser posicionadas 4 inimigos no labirinto. As habilidades dos inimigos são as seguintes:

Sondagem: inimigos podem sondar até 7 posições em linha reta, assim como exibido abaixo. Dessa forma, se o player entrar na área de sondagem do inimigo a posição atual do player passa a ser o alvo do inimigo, ou seja, ao detectar o player, o inimigo marca a posição atual do player como seu alvo, e nas próximas jogadas, o inimigo se desloca até o alvo marcado. Ao chegar na posição marcada, o inimigo deve aplicar o processo de sondagem novamente no sentido de identificar um novo alvo. A perseguição termina quando o inimigo capturar o player ou quando o player sair da área de sondagem do inimigo.

'#'	(7) ↑	'#'
7 posições ←		→ (7)
'#'	↓ (7)	'#'

Salto: Por padrão, um inimigo pode avançar no labirinto saltando 1 posição por vez. Contudo, um inimigo pode se fundir com outro sempre que uma colisão for detectada. Basicamente uma colisão irá acontecer sempre que dois ou mais inimigos ocuparem a mesma posição. O resultado de uma colisão é um inimigo com maior capacidade de salto. Portanto, dada uma colisão, a capacidade de salto do inimigo resultante é a soma das capacidades de salto de todos os inimigos envolvidos na colisão.

Símbolos utilizados

O aluno pode utilizar a tabela abaixo para representar os elementos do jogo:

Elementos	Interface padrão
Player	'X'
Inimigo padrão (salto = 1)	'1'
Inimigo fundido (salto > 1)	'2', '3', '4'
Saída	'S'
Caminho	'.' (ponto)
Parede	'#'

Dimensões do jogo

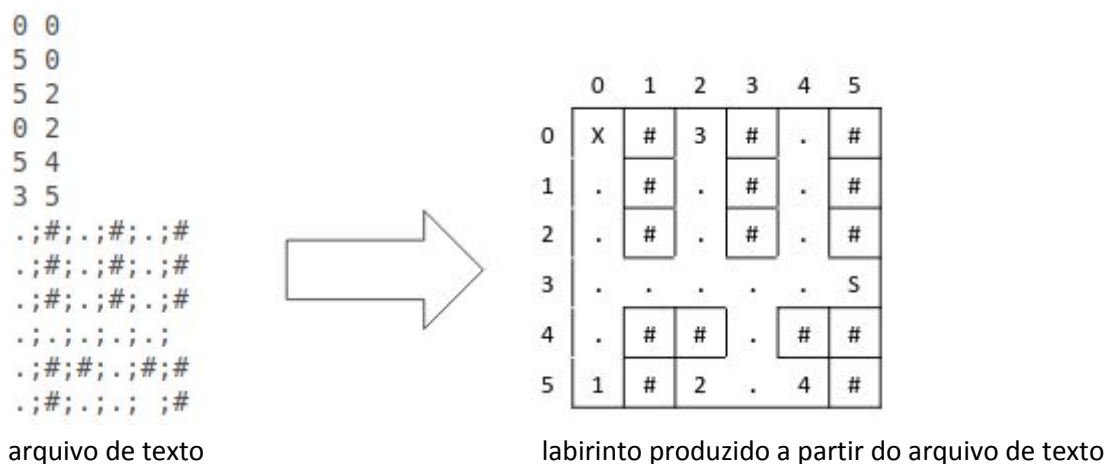
O labirinto será implementado por uma matriz com dimensões fixas. Em particular, a matriz utilizada para representar o labirinto deverá ser composta por 21 linhas e 21 colunas.

Arquivo de entrada

O jogo deve ser carregado a partir de um arquivo de texto. Em particular, esse arquivo de texto deve ser produzido a partir do arquivo “gerador_labirinto.c” disponível no Moodle. O labirinto descrito no arquivo de texto deve ser utilizado para criar a estrutura lógica (matriz) que irá representar o labirinto. Seu arquivo de texto deve respeitar a seguinte estrutura (NÃO SERÃO CONSIDERADAS ESTRUTURAS DIFERENTES):

Linha 1: x y (posição inicial do player)
Linha 2: x y (posição inicial do inimigo 1)
Linha 3: x y (posição inicial do inimigo 2)
Linha 4: x y (posição inicial do inimigo 3)
Linha 5: x y (posição inicial do inimigo 4)
Linha 6: x y (posição da saída do labirinto)
Linha 7: status de todas as colunas da primeira linha da matrix (parede '#' ou caminho '.') Use o caractere ponto e vírgula (';') como separador de conteúdo
...
Linha 27: status de todas as colunas da última linha da matrix (parede '#' ou caminho '.')

Exemplo: convertendo arquivo de texto para labirinto (matriz 6 x 6):



Principais atributos de cada elemento do jogo (TADs)

Maze:

- int: número de linhas
- int: número de colunas
- Position [][]: matriz de jogo

Position:

- char: símbolo que será exibido ('#', '.', 'X', '1', '2', '3' ou '4')
- int: x (uma linha da matriz)
- int: y (uma coluna da matriz)
- int: rastro (opcional)

Player:

- char: símbolo que será exibido ('X')
- int: status (vivo/morto)
- Position: posição inicial do player
- Position: posição atual do player

Inimigo:

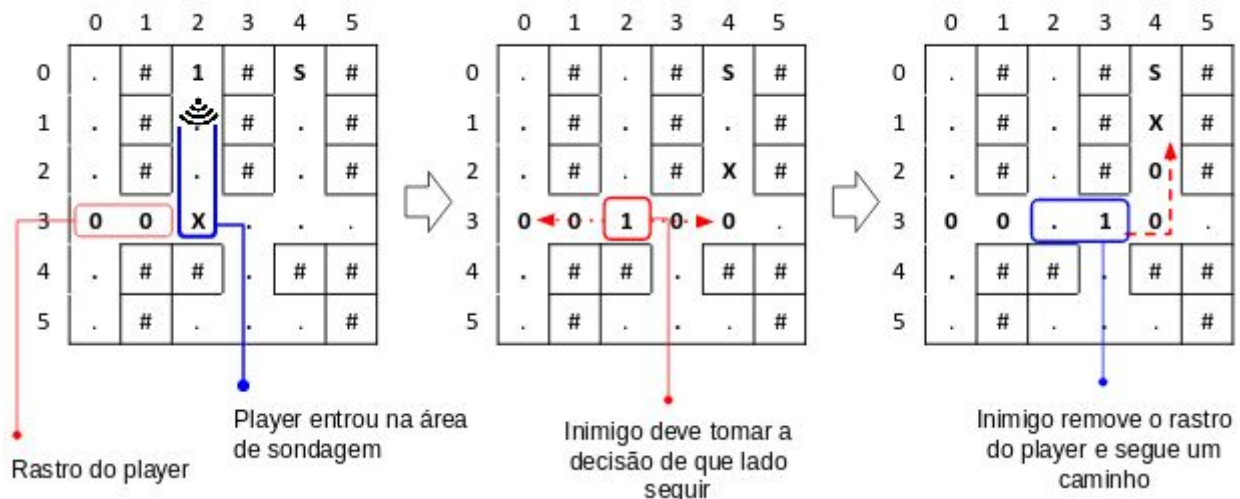
- char: símbolo que será exibido ('1', '2', '3' ou '4')
- int status (vivo/morto)
- int: capacidade salto
- Position: alvo (última posição percebida do player)
- Position: posição inicial do inimigo
- Position: posição atual do inimigo

Saída:

- char: símbolo que será exibido ('S')
- Position: posição da saída na matriz

Ponto extra:

Perseguição por rastro: utilize o campo rastro do TAD *Position* para implementar essa funcionalidade. Basicamente, nessa abordagem, o player marca cada célula da matriz que visita. Dessa forma, quando um inimigo detecta o player como alvo, esse inimigo persegue o player a partir do rastro e não mais pela sondagem, tal como exibido no exemplo abaixo. Note que a solução exibida abaixo foi pensada para apenas um único inimigo. Portanto, o ponto extra consiste em elaborar um sistema de rastro que funcione para todos os 4 inimigos.



Obs 1.: A modelagem do projeto deve ser feita utilizando os TADs, tal como apresentado em sala de aula.

Obs 2.: O trabalho pode ser feito no máximo em dupla.