MapModule

O MapModule foi originalmente formado pelo Henrique Noronha Facioli e pelo Thiago Silva de Farias e tinha como objetivo o gerenciamento de todos os eventos que ocorrem no mapa do jogo (Corredor). O MapModule deve criar uma matriz de iCells, gerenciar o que acontece e o alcance de visão do jogador no mapa.

Inicialmente ele deve posicionar os iPuzzles, utilizando um algorítimo aleatório (Java.Random) nas paredes do mapa, e garantir que elas sempre estejam no mesmo local. Deve, também, posicionar os iMob's presentes no jogo (monstros) de uma maneira aleatória, instanciando eles para aumentar a dificuldade conforme a posição da celula na matriz e, por fim, posicionar o jogador principal em alguma célula da matriz definida estaticamente.

Para que seja garantido que o mapa seja o mesmo durante uma partida, decidimos usar como design pattern o Singleton, fazendo assim que, mesmo que o jogador entre em algum modo de jogo que não o corredor, quando voltar a ele, volte todas as configurações do jeito que ficaram e também não tenha perigo de se instanciar duas vezes o mapa.

Durante o jogo, o MapModule deve gerenciar as colisões entre iMob's e entre iMob e iRoom. Caso dois iMobs se encontrem ele deve verificar se são monstros, e se forem apenas ignorar; Caso seja o personagem principal e um monstro, deve retornar a string de "BattleMode" para que o ManagerModule possa acionar este módulo; Caso seja um iMob e iRoom, deve verificar se é o personagem principal e se sim, retornar a string "RoomMode" para que o ManagerModule possa acionar o módulo necessário.

Especificações

Interface iMap

- O método “move” recebe um ‘iMob’ e uma direção na forma de ‘char’, e retorna um ‘boolean’. Assim, caso o ‘iMob’ seja movido na direção indicada com sucesso, retornamos ‘true’, e, caso contrário, retornamos ‘false’.
- O método “getMap” retorna, a partir do campo de visão do personagem, uma string contendo o mapa que é possível observar a partir do ponto em que o jogador está.

Classe Map

- Os atributos da classe Map são: ‘TAM_X’ e ‘TAM_Y’, que são do tipo ‘static int’, com a finalidade de determinar o tamanho do mapa; ‘instance’, que é do tipo ‘Map’ e é utilizada no método de ‘Singleton’ especificado abaixo; uma matriz de ‘iCells’, que representa o corredor.
- Os métodos “move” e “getMap” simplesmente implementam os métodos da interface, especificados acima.
- O método “getInstance” é um ‘Singleton’, garantindo que o mapa será implementado apenas uma vez. Ele recebe um ‘iMob’, que representa o ‘player’, e, caso o mapa ainda não tenha sido instanciado, cria um mapa com o construtor especificado abaixo. Caso contrário, ele simplesmente retorna o mapa que já havia sido criado.
- O construtor do mapa, ‘Map’, é do tipo privado. Ele recebe um ‘iMob’, que representa o ‘player’, e então ele o posiciona, além de determinar posições para os ‘iPuzzles’ e os montros, no formato ‘iMob’. Em ambos os casos, a escolha para as posições será randômica, portanto fazemos uso do ‘Java.Random’.

CellModule

O CellModule foi originalmente criado separado do MapModule e o grupo encarregado é o do Lucas Silva Schanner e do Lauro Cruz e Souza. Com o andamento do projeto percebemos que o CellModule e o MapModule eram intrínsecos e não dava para separar, e portanto acabamos unindo os grupos. O CellModule deve conter as células e as posições.

As células podem armazenar em si um jogador ou um monstro (Mob) ou uma porta (Puzzle). Cabe a célula realizar as operações de retirar e adicionar em si mesma esses objetos.

Especificações

Classe Cell

- Método removeMob: retira o Mob presente na célula (se houver um) e retorna esse mob retirado.
- Método setMob: coloca um Mob na celula (player ou monstro) e retorna true se foi possível colocá-lo e false se não foi (ou seja, já há um mob ou uma porta na célula)
- Método getMob: Retorna o Mob presente na célula, retorna null caso não tenha nenhum.
- Método getDoor: Retorna a porta IPuzzle presente na célula. Retorna null caso nao tenha nenhuma.
- Método setParede: Define se a célula é uma parede, recebendo como parâmetro um boolean.
- Método getParede: retorna se a célula e ou nao uma parede (true ou false).
- Método getDescription: retorna um caracter que representa o que existe naquela célula. Usado para imprimir o mapa na tela.
- Atributos privados que armazenam o Mob (IMob), a porta (IPuzzle) e se é ou não parede (boolean).

bdModule

Com a união do CellModule e do MapModule, separamos os grupos e decidimos criar o bdModule que ainda está em fase embrionária. Ele será responsável pela criação, leitura e remoção de informações em algum banco de

dados. Estamos estudando a possibilidade de usar JDBC com mySQL ou criarmos o nosso próprio utilizando algum formato próximo do CSV.

[PDF DO SANTANCHE]
-http://www.ic.unicamp.br/~santanch/teaching/oop/2015-1/trabalho/trabalho-05-mc302-2015-1-v01.pdf