Criado em 1989 por Milton Bradley, o jogo Hero Quest é um RPG (role playing game) de aventura. Nesse projeto, nós Maicon Gabriel de Oliveira, RA 221329, Murilo Gomes da Silva, RA 242233 e Tiago Feliciano Gomes, RA 224956 tivemos como objetivo produzir uma adaptação simplificada do jogo seguindo os critérios aprendidos durante a matéria de programação orientada a objetos na linguagem Java. Nosso projeto conta com as seguintes funcionalidades:

**Carregamento de Mapa**

**Montagem de cenário pré-definido:** Através de um arquivo de texto com um mapa de caracteres no disco, o jogo carrega o arquivo e gera o cenário de acordo com o que foi especificado no arquivo, como o as dimensões do tabuleiro, quantidade de andares na torre e disposição de itens como monstros(tipo e localização inicial no mapa), portas, armadilhas, baús, portas ocultas e paredes.

**Montagem de cenário aleatória**: Herdando o princípio de carregar a disposição das paredes de um arquivo de texto no disco e 3 modelos pré-definidos possíveis para geração de portas no cenário, os demais itens como disposição de portas ocultas, localização e tipo de monstros, existência ou não de armadilhas, localização de baús e sua disposição no mapa são gerados aleatoriamente no código.

**Portas na montagem de cenário aleatória:** Devido à dificuldade em garantir que todas as salas tenham portas, para a criação de portas comuns no cenário, foram dispostas 3 possibilidades de disposição de portas previstos no arquivo de carregamento das paredes, sendo assim escolhido aleatoriamente um dos conjuntos de portas na montagem de cenário aleatória. Para a criação de portas ocultas, foi definido que elas devem ser geradas aleatoriamente seguindo os seguintes critérios: deve substituir uma parede e, a parede deve ter 2 de seus lados opostos livres.

No carregamento do cenário foi definido por padrão que os personagens principais (heróis incluindo o jogador) sempre serão carregados nas posições adjacentes a posição central do mapa, e, fica reservado a posição central como uma possível posição de escada para ir ao próximo andar (o último andar não possui escada pois não é necessário).

**Componentes de cenário**

Foi decido que todos os objetos do mapa herdam de uma classe GameObject, a qual estabelece alguns atributos essenciais a todos os componentes do mapa, são estes: nome, ID (identificação específica do item), Coordinate (posição no mapa) e Sprite (representação ao renderizar cena). Assim como métodos para acessar indiretamente tais atributos e redefinir os tipos nome e Coordinate.

Foi definido inicialmente que todo mapa pode conter os seguintes objetos:

**Heróis:** Bárbaro (B), Anão (D), Elfo (E) e Mago (W).

**Monstros:** Goblin (G), Esqueleto (K) e Esqueleto Mago (M).

**Estruturais:** Wall (#), Door (U), Hidden Door (# -> U) e Stair (S).

**Mobília:** Armadilha\* e Baú (C).

Todos os objetos especificados são intransponíveis, exceto a armadilha, a qual ao ser transposta, é ativada, causa dano e é removida do mapa.

**Especificação do funcionamento dos componentes de cenário:**

**Heróis:** Todos os heróis realizam uma ação e após se deslocam. Para o deslocamento, todos os heróis (exceto o controlado pelo player) escolhem uma direção aleatoriamente e andam um número de passos também escolhidos aleatoriamente desde que sejam até o valor máximo obtido nos dados. Todos realizam ataque corpo-a-corpo e, no caso do Elfo e Mago também podem lançar os feitiços que tiverem equipados.

**Monstros:** Todos os monstros realizam uma ação escolhida aleatoriamente entre atacar ou lançar uma magia, caso o monstro não possua habilidade para lançar magias, sempre realizam a ação de atacar. Após feito a ação, todos se movimentam da seguinte forma:

* Goblin: Escolhe um lado baseado na distância do herói mais próximo, após rola os dados de deslocamento para definir a distância máxima, e, novamente aleatoriamente escolhe uma quantidade de passos dentro da distância máxima.
* Esqueleto: Escolhe aleatoriamente um lado para locomoção, rola os dados de distância máxima, e, dentro desse limite anda aleatoriamente um certo número de casas.
* Esqueleto Mago: Após atacar corpo-a-corpo ou lançar magia, escolhe aleatoriamente um lado para locomoção, rola os dados de distância máxima, e, dentro desse limite anda aleatoriamente um certo número de casas.

**Door (U):** Quando o jogador interage com método “usar” adjacente a ela, seu objeto é apagado, liberando o caminho permanentemente

**Hidden Door (# -> U):** Sendo uma extensão de Door, quando o jogador interage com o método “procurar” adjacente a esta, ela passa a se comportar como uma Door comum.

**Stair (S):** Quando o jogador interage com método “usar” adjacente a ela, caso não existam mais monstros no andar, é chamado o mapa do andar seguinte e os heróis (que estiverem vivos) são movidos para o centro do mapa seguinte.

**\*Armadilha (T):** Inicialmente pensado como algo móvel (como armadilha para animais) e por isso classificado como “forniture”, é sempre oculta na renderização. Ao utilizar método “procurar” em sua adjacência, o jogador é notificado e o item armadilha é apagado, caso contrário, caso algum herói tente transpor ela, é ativada causando um dano pré-estabelecido no herói. Para que monstros não ativem a armadilha, foi estabelecido que monstros não se movem para cima de armadilhas, pois uma vez que tem conhecimento sobre estas, as evitam.

**Baú (C):** Uma vez aberto pelo jogador, este recebe um item aleatório do jogo (do tipo CanCarry pois pode ser carregado no inventario) e depois tem seu objeto apagado do mapa.

**Classes auxiliares**

**Sprite –** Classe que tem como função representar a saída(imagem) dos objetos na renderização, possui métodos para alternar facilmente entre as possíveis imagens que um objeto pode ter, como animações ou representações de um objeto oculto utilizando a sprite de outro objeto. Foi estabelecido que a Sprite de todo objeto inicialmente é oculta na renderização e, uma vez que o player a tenha em uma área visível, essa passa a exibir a verdadeira sprite do objeto.

**Coordinate –** Classe que tem como finalidade representar a posição de um objeto no mapa, possui métodos para definição, acesso, redefinição de coordenadas e cálculo da distância entre duas posições (usado para que um personagem possa seguir o outro. Ex: Goblin).

**StartEquipment –** Classe que tem como objetivo definir e retornar o equipamento inicial de cada herói assim como seu conjunto inicial de magias.

**EquipmentLoad –** Classe que tem como objetivo pré-carregar todos os itens coletáveis do mapa para disponibilizá-los aleatoriamente na procura de itens pelo mapa, baús e loja do jogo.

**MonstersLoad –** Classe que tem como objetivo definir e retornar o equipamento inicial e magias de cada monstro do jogo.

**Money –** Classe que tem como objetivo armazenar e estabelecer relações com o dinheiro do jogo, como métodos para adição, remoção e administração de montantes.

**Enumeradores**

* **SpellTypes -** Define os tipos de feitiços existentes no jogo (suporte, ataque, ataque em area).
* **SpellElements –** Define os possíveis elementos de uma magia (Água, Terra, Ar e Fogo).
* **ArmorClasses –** Define os tipos de objetos de defesa existentes no jogo (Armadura leve, Armadura pesada e Escudo) e estabelece e verifica quais personagens podem equipar cada tipo de item de defesa.
* **WeaponTypes** **–** Define os tipos de armas(espadas) existentes no jogo (Adaga, Espada curta ou Espada longa) e estabelece e verifica quais personagens podem equipar cada tipo de arma.
* **WhiteDiceSides –** Define os lados de um dado branco de combate.
* **GameTypeObjects –** Possui a identificação de todos os objetos contidos no jogo, estabelece e faz a verificação se um personagem é ou não usuário de magias.
* **Command –** Define todas as ações que um personagem pode desempenhar, como direções de movimento, comando “procurar”, “usar”, “atacar” e “usar magia”. Adicionalmente pode retornar uma direção aleatoriamente dentre “cima”, “baixo”, “direita” e “esquerda”.
* **PlayableClasses –** Define as classes jogáveis do jogo, isto é, a classe dos heróis (Bárbaro, mago, elfo ou anão).
* **MapMode –** Define o tipo de carregamento de mapa que deve ser tomado (predefinido ou aleatório).
* **GameMode –** Define as possíveis dificuldades do jogo, isto é, normal ou difícil (Afeta a quantidade de armadilhas ocultas no mapa gerado aleatoriamente e o dano que cada uma provoca, além de, ao encontrar monstros utilizando a função “Procurar”, definir a quantidade que será encontrada destes).

**Exceções**

* **FullInventoryException-** Exceção lançada quando o inventário do jogador está cheio e ele tenta adicionar alguma coisa.
* **InvalidOperationException –** Exceção lançada quando o jogador tenta realizar uma operação inválida.
* **InvalidTypeException –** Exceção lançada quando o usuário tenta criar um objeto com um tipo inválido.

**Funcionalidades adicionais**

* Market – Entre os andares da torre, é disponibilizado uma loja para compra e venda de itens coletados (possui todos os equipamentos disponíveis do jogo, poções de cura e magias). A loja segue as seguintes premissas:
  + A venda de um item sempre tem seu valor decrescido em 5POs.
  + Magias são tratadas como livros de encantamentos, seu uso é ilimitado e múltiplos livros podem ser adquiridos, porém seu valor de adesão é alto, e, dentre os heróis apenas o Elfo e o Mago podem utilizá-las.
  + Para a venda de itens a loja é necessário que a loja tenha fundos para comprar tal item.
* Escolha de nome personalizado para o jogador.
* Uso de poções para restaurar a vida do jogador (basta utilizar comando “Usar item” não estando na adjacência de nenhum objeto interagível) e ter ao menos uma “Health Potion” no inventário.
* Possibilidade de existência de vários andares na torre (especificado no arquivo de texto carregado), a troca de andar pode ser feita interagindo com o a escada “S” após derrotar todos os monstros do andar.
* Existência de escudos para o aumento de dados de defesa.
* Existência de baús contendo itens aleatórios pelo mapa.
* Caso a função procurar não encontre uma armadilha ou porta secreta, essa tem chance de encontrar novos monstros, dinheiro, equipamentos, magias ou nada.
* Auto equipagem de itens, ao adicionar item equipável ao inventario, caso seja melhor que o equipado, faz a troca entre os dois. No caso de armas de uso único, após remover a arma usada procura no inventario pela próxima melhor arma e a equipa.

**Equipamentos**

Todos os personagens do jogo (heróis e monstros) possuem um inventario no qual podem carregar itens, é definido como um item que pode ser carregado todos os itens que herdam da classe CanCarry, sendo assim dinheiro, espadas, escudos, armaduras, magias e consumíveis como poções de cura. Ficou pré-estabelecido que o número de itens que cada personagem pode carregar se limita a 30 itens não empilháveis (desconsiderando os itens equipados pelos personagens).

Cada personagem possui 4 slots de equipamentos, sendo 2 para equipamentos de defesa(armadura e escudo) e 2 para equipamentos de ataque, assim é possível diversas combinações de equipamentos, lembrando que, caso equipado uma espada de 2 mãos, o escudo e/ou segunda espada equipada é movida para o inventário automaticamente, assim como equipar uma segunda espada remove o escudo equipado e vice-versa. A equipagem dos equipamentos e magias é feita de forma automática pelo jogo, dando sempre prioridade em manter equipado os itens de maior ataque existentes no inventário. Foi definido as seguintes restrições segundo os personagens:

* Bárbaro: Pode equipar qualquer arma, armadura ou escudo do jogo.
* Anão: Pode equipar armaduras leve, escudo e espadas curtas ou adagas.
* Elfo: Pode equipar armaduras leves, espadas curtas ou adagas.
* Mago: Não pode equipar itens de defesa, mas pode equipar adagas.
* Esqueleto: Pode usar qualquer arma do jogo.
* Goblin: Pode equipar adagas.

Equipamentos disponíveis:

**Armamento:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nome | Tipo | Uso único | Dano(dado) | Alcance | Observação |
| Dagger | Uma mão | Sim | 1 | 2 | Inicial do Mago(x3) |
| Better Dagger | Uma mão | Sim | 3 | 3 | Inicial do Elfo e Anão |
| Short Sword | Uma mão | Não | 2 | 1 |  |
| Better Short Sword | Uma mão | Não | 4 | 1 |  |
| Long Sword | Duas mãos | Não | 3 | 1 | Inicial do Bárbaro |
| Better Long Sword | Duas mãos | Não | 4 | 2 |  |

**Defesa:**

|  |  |  |
| --- | --- | --- |
| Nome | Classe | Defesa(dado) |
| Light Armor | Armadura leve | 1 |
| Better Light Armor | Armadura leve | 2 |
| Heavy Armor | Armadura pesada | 3 |
| Better Heavy Armor | Armadura pesada | 4 |
| Shield 1 | Escudo | 1 |
| Shield 2 | Escudo | 2 |

**Cura:**

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Tipo | Uso único | Descrição |
| Health Potion | Cura | Sim | Recupera 4 pontos de vida do jogador |

**Feitiços**

Personagens com afinidade a magia como Magos, Elfos e Esqueletos Mago, desde que possuam a magia “equipada” podem lançar feitiços em seus oponentes ou em si mesmo. Dividido entre magias de “suporte”, “ataque” e “ataque em area” cada magia pode ser usada infinitamente, porém deve ser checado se o lançador consegue realizar o feitiço antes de cada ataque, e, apesar de todos com afinidade poderem utilizar todos os tipos de magias, seu custo de aquisição é mais alto do que de itens comuns. São magias disponíveis:

* Teleport: Move o usuário da magia para outra posição visível por ele. O movimento deve ser feito a sul, norte, leste ou oeste, e, não deve haver obstáculos no meio do caminho.
* Simple Heal: Cura um d6 de vida do lançador da magia.
* Magic Missile: Caso haja algum oponente na distância de 3 casas do lançador, ataca o oponente mais próximo com 3 flechas magicas causando 2 danos cada.
* Fireball: Caso haja algum oponente na distância de 2 casas do lançador, ataca o mais próximo causando danos em area com uma bola de fogo que causa 6 danos no alvo e, caso haja inimigos adjacentes na distância de 1 casa do alvo, esses sofrem 3 danos.

**Jogabilidade**

O jogo consiste em turnos onde o jogador realiza uma ação e se locomove. Antes do início do jogo, o jogador deve escolher uma dificuldade dentre “NORMAL” ou “DIFICIL”, se deseja jogar em um mapa pré-carregado ou gerado aleatoriamente e, qual classe dentre os heróis disponíveis vai incorporar (Bárbaro, Elfo, Anão ou Mago), implicando na possibilidade ou não de lançar feitiços durante o gameplay e quais deles estarão disponíveis inicialmente.

O ato de andar consiste em indicar uma direção de movimento (W – norte, A – oeste, S – sul, D - leste) e, após o sistema rolar dois dados de 6 faces para indicar a distância máxima que o jogador pode se locomover, escolher uma quantidade de passos que o jogador pretende andar. Monstros e demais heróis tem seu movimento controlado pelo jogo.

O ato de realizar uma ação se distingue pelo tipo de personagem que a realiza, o jogador, por padrão pode realizar ações de “Procurar”, “Usar item”, “Atacar”(ataque corpo-a-corpo) e, caso escolhido a classe Elfo ou Mago pode utilizar a ação de “Usar magia”. Os demais heróis e monstros não possuem as ações “Procurar” e “Usar item”, e, “Atacar” e “Usar magia” é escolhido aleatoriamente pelo sistema.

O jogo tem como objetivo eliminar todos os monstros presentes em todos os andares, assim, o jogo é tido como ganho caso não haja mais nenhum monstro em nenhum andar do jogo e, é tido como perdido caso o jogador morra, lembrando que a torre pode possuir vários andares e, que a escada para o próximo só é liberada caso não haja mais monstros no andar atual.

**Ações**

**Procurar –** Busca na adjacência do jogador por armadilhas (e as desarma) ou portas secretas, caso não haja nenhum dos dois, possui uma chance de 50% de encontrar novos monstros (1 a 3 na dificuldade NORMAL e 3 a 6 na dificuldade DIFICIL) gerados em posições aleatórias no andar atual, 25% de chance de não encontrar nada, 12,5% de chance de encontrar uma quantia em dinheiro variando entre 5 e 50POs (peças de ouro) e 12,5% de chance de encontrar um item aleatório dentre armas, armaduras, feitiço ou poção de cura.

**Usar item –** Busca na adjacência do jogador por itens interagíveis (Portas, baús ou escada), caso encontre, realiza a interação, caso não haja nenhum destes, pergunta ao jogador se quer interagir com o inventário tomando uma poção de cura, se sim, e houver poções de cura no inventário o jogador tem sua vida recuperada dentro dos limites pré-estabelecidos.

**Atacar –** Busca por oponentes nas posições adjacentes ao atacante no alcance da arma de maior alcance equipada, caso encontre, realiza dano corpo-a-corpo como especificado nas instruções do projeto.

**Usar magia –** Caso o atacante possa fazer o uso de magias, rola dado para verificar se o atacante consegue soltar a magia e, caso sim, aplica a magia sobre si mesmo no caso de magias como “Teleport” ou “Simple Heal”, ou, realiza ataque magico no(s) oponentes dentro do raio de alcance da magia.

**Diagrama UML das principais classes**