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 dispostos 3 possibilidades de disposição de portas previstos no arquivo de carregamento das paredes e, 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).

**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:

**Herois:** Barbarian, Dwarf, Elf e Wizard.

**Monstros:** Goblin, Skeleton e Mage Skeleton.

**Estruturais:** Wall, Door, Hidden Door e Stair.

**Mobília:** Trap e Chest.

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

**Especificação dos componentes de cenário:**

**Herois:** Todos os heróis (exceto o escolhido pelo player) escolhem uma direção aleatoriamente e andam um número de passos também escolhidos aleatoriamente entre 1 e o máximo obtido nos dados. Todos realizam ataque corpo-a-corpo e, no caso do Elf e Wizard também lançam feitiços.

**Monstros:**

* Goblin: Escolhe um lado baseado na distância do herói mais próximo, a cada turno o persegue e ataca corpo-a-corpo.
* Skeleton: Escolhe um lado para locomoção, anda aleatoriamente e ataca corpo-a-corpo.
* Mage Skeleton: Escolhe um lado para locomoção, anda aleatoriamente e lança feitiços.

**Door:** Quando o jogador interage com método “usar”, seu objeto é apagado, liberando o caminho permanentemente

**Hidden Door:** Sendo uma extensão de Door, quando o jogador interage com o método “procurar” adjacente a esta, ela se transforma em uma Door comum.

**Stair:** Quando o jogador interage com método “usar”, caso não existam mais monstros no andar, é chamado o mapa do andar seguinte e os heróis são movidos para o centro do mapa.

**Trap:** Inicialmente pensado como algo móvel (como armadilha para animais), é sempre oculta na renderização. Ao utilizar método “procurar” em sua adjacência, tem seu objeto apagado, caso contrário, caso algum herói tente transpor ela, é ativada causando um dano pré-estabelecido no herói. Obs: 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.

**Chest:** Uma vez aberto pelo jogador, este recebe um item aleatório do jogo 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.

**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).

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

**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.

**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)
* **WeaponTypes** **–** Define os tipos de armas(espadas) existentes no jogo (Adaga, Espada curta ou Espada longa)
* **WhiteDiceSides –** Define os lados de um dado branco de combate.
* **GameTypeObjects –** Possui a identificação de todos os objetos contidos no jogo
* **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.
* **PlayableClasses –** Define as classes jogáveis do jogo, isto é, a classe dos heróis.
* **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.

**Exceções**

* **FullInventoryException-** Exceção lançada quando o inventário do jogador está cheio e ele tenta adicionar alguma coisa.
* **InsufficientFundsException –** Exceção lançada quando o jogador não possui saldo suficiente para fazer uma compra.
* **InvalidOperationException –** Exceção lançada quando o jogador tenta realizar uma operação inválida.
* **InvalidTypeException –** Exceção lançada quando o usuário tenta criar um objeto com um tipo inváldido.

**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 e poções de cura).
* Escolha de nome personalizado para o jogador
* Uso de poções para restaurar a vida do jogador (basta utilizar comando “usar” não estando na adjacência de nenhum objeto intragável)
* Possibilidade de existência de vários andares na torre (especificado no arquivo de texto carregado)
* Existência de escudos para aumento de status
* Existência de baús contendo itens aleatórios pelo mapa

**Equipamentos**

Todos os personagens do jogo (heróis e monstros) possuem 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 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 é feito de forma automática pelo jogo, dando sempre prioridade em manter equipado os itens de maior ataque existentes no inventário. Inicialmente não foram estabelecidas restrições de classes quanto ao uso de equipamentos.

**Jogabilidade**

Antes do início do jogo, o jogador deve escolher 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.

O jogo consiste em turnos onde o jogador se locomove e realiza ou não uma ação a cada tileset por onde passa.

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 player pode se locomover, escolher uma quantidade de passos que o jogador pretende dar (entre 1 e máximo rolado). 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”, “atacar”(ataque corpo-a-corpo) e, caso escolhido a classe Elfo ou Mago pode utilizar a ação de “lançar feitiço”. Os demais heróis e monstros não possuem as ações “procurar” e “usar”.

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 um novo monstro gerado em posição aleatória no andar atual, 25% de chance encontrar uma quantia em dinheiro variando entre 5 e 50POs (peças de ouro) e 25% de chance de encontrar um item aleatório que pode ser carregado pelo jogador.

**Usar –** 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, caso encontre, realiza dano corpo-a-corpo como especificado nas instruções do projeto.

**Lançar Feitiço –** Busca por oponentes em torno jogador seguindo o raio de alcance do feitiço, caso encontrado, realiza dano mágico como especificado nas instruções do projeto.