

Programação Orientada por Objectos (POO)

Semestre de verão de 2010-2011

Trabalho Final

LEETC – LEIC – LERCM

OBJECTIVOS

Neste trabalho os alunos adquirem prática na utilização dos recursos existentes na biblioteca JAVA para a criação de aplicações com interface gráfica e consolidam os temas tratados na unidade curricular.

DATAS:

04 de Julho de 2011 - Esboço do diagrama estático de classes UML.

18 de Julho de 2011 - Entrega de uma versão funcional do trabalho.

A data de entrega da versão final, e do relatório, será combinada com o docente de cada turma.

DESCRIÇÃO

Desenvolver uma aplicação em Java que implemente o jogo *Rabbit Shoot*.

O jogo *Rabbit Shoot* consiste em abater **coelhos**. É um jogo por níveis e em cada nível o jogador dispõe de um determinado tempo para abater cinco coelhos que se movimentam no **terreno**. O número de pontos, em cada nível, é proporcional aos coelhos abatidos e inversamente proporcional ao tempo despendido.

No **terreno**, área rectangular em que se confina a acção, existem **coelhos** brancos e cinzentos; **tocas** que simulam entradas e saídas em caminhos subterrâneos; **arbustos** que permitem refúgio; e **pedras** intransponíveis. Aleatoriamente, no tempo e no espaço, aparecem no terreno **bombas** que explodem destruindo todos elementos existentes numa dada região, ou **bónus** que permitem que o jogador ganhe pontos ou tempo.

Quando o **jogador** dispara a arma, abate coelhos; explode bombas; ou adquire bónus, consoante o elemento que se encontre sob a mira. O jogador orienta a **mira** da arma e dispara através do *mouse*.

Os **coelhos** deslocam-se no terreno a uma determinada velocidade em função do nível, evitando as pedras, entrando e saindo das tocas e refugiando-se nos arbustos. O **coelho branco** segue uma trajectória horizontal, vertical ou oblíqua e só muda de sentido quando colide com as pedras ou quando no caminho lhe aparece uma toca. O **coelho cinzento** tem sempre como objectivo esconder-se numa toca. Quando um coelho é abatido nasce outro, da mesma cor, numa posição aleatória válida.

A figura ilustra um possível estado do jogo, o aspecto gráfico é um mero exemplo, que não é obrigatório seguir.



Deverá estar afixada durante todo o jogo, a informação relativa à pontuação, ao número de coelhos a abater e ao tempo que falta para terminar o nível.

O jogo termina ao passar o último nível ou se o jogador não conseguir abater os cinco coelhos dentro do tempo limite.

Níveis

Devem ser implementados no mínimo **dois níveis**. O tempo de cada nível é de um minuto mais o somatório dos tempos de bónus adquiridos no nível. O jogo muda de nível quando o jogador abate cinco coelhos dentro do tempo. Ao passar para um novo nível de jogo, é mudado o cenário e iniciado o tempo de duração do nível para um minuto. O cenário correspondente a cada nível é obtido de um ficheiro de texto.

O primeiro nível tem dois coelhos, um branco e um cinzento, arbustos, pedras, tocas, bem como bombas (no mínimo de dois tipos) e bónus que devem aparecer de forma aleatória. O segundo nível tem o dobro dos coelhos dos arbustos e das tocas, e os coelhos devem deslocar-se a maior velocidade.

Pontuação

A *pontuação do jogo* é calculada em função da quantidade de coelhos abatidos, dos pontos de bónus recolhidos e do tempo em que cada nível é concluído (quanto mais rápido, mais pontos). A interface gráfica deve apresentar a pontuação do jogador durante o decorrer do jogo.











Em cada nível, deve ser mantido o *registo das dez pontuações máximas*, com o respectivo nome do jogador. O jogador, em qualquer momento, pode consultar essa informação.

[OPCIONAL]

Funcionalidades adicionais:

- pausa *on-off*.
- *memorização do estado de um jogo*, de modo a que o utilizador possa recomeçar a jogar a partir do estado guardado. Quando o jogo se inicia, caso exista jogo memorizado, o jogador será interrogado se deseja ou não continuar o jogo.

Tabela descritiva dos elementos:

Elementos do cenário	Descrição
	Pedra. Quando o coelho colide com este elemento muda de trajectória.
	Buraco. Quando o coelho entra numa toca a sua saída ocorre noutra toca.
	Terra. Define o chão do jogo.
	Arbusto. Refúgio do coelho. Quando o coelho se desloca atrás dos arbustos não é abatido.
Bombas	Descrição
	Bomba1. O raio de acção é uma linha completa.
	Bomba2. O raio de acção é uma área de 5 por 5 quadrículas.
	Chama (flame). Efeito produzido durante a explosão das bombas em todo o raio de acção.
Actores	Descrição
	Mira da arma do Jogador. Move-se em qualquer direcção cardinal através do rato.
	Coelho branco. Cada coelho abatido vale 10 pontos. Desloca-se a uma dada velocidade dentro do terreno podendo refugiar-se nos arbustos e nas tocas.
	Coelho cinzento. Cada coelho abatido vale 20 pontos. Desloca-se a uma dada velocidade dentro do terreno na direcção das tocas podendo refugiar-se nos arbustos. Quando não existirem tocas (foram entretanto destruídas pelas bombas) tem o comportamento igual ao do coelho branco.
Bónus	Descrição
	Pontos.
	Tempo.

ENTREGA DO TRABALHO

Elabore um esboço do diagrama estático de classes UML que representa os vários elementos que compõem o jogo e as relações entre eles. Antes de proceder à implementação, discuta as soluções encontradas com o docente. Tenha em consideração a utilidade de separar a interface gráfica da lógica do jogo. A interação com o docente tem como data limite **4 de Julho**.

A data de entrega de uma versão funcional do trabalho é **18 de Julho**. A data de entrega da versão final e do relatório será combinada com o docente de cada turma.

O relatório do trabalho deve evidenciar quais dos temas estudados ao longo do semestre foram aplicados e quais as vantagens resultantes da sua aplicação na implementação do jogo. Em particular, os seguintes temas: algoritmos recursivos; herança e polimorfismo; tratamento de exceções; *streams*; estruturas de dados; e programação *event driven*.

Na concepção da solução note que se valoriza:

- A qualidade do código produzido (facilidade de alteração; não repetição de código; modularização; documentação);
- As estruturas de dados e os algoritmos utilizados;
- A facilidade de utilização da aplicação;
- Criatividade.

Bom trabalho e seja criativo.
Os docentes de POO