Enunciado do projeto de POO 2021/22 (v1.0)

FireFight

Introdução

Neste projeto pretende-se criar um jogo que simula o combate a incêndios. O jogo joga-se sobre uma grelha. Os jogadores controlam um bombeiro que tenta apagar os fogos. O jogo é *turn-based*, o que significa que apenas acontece alguma coisa quando se carrega numa tecla.

A cada posição da grelha de jogo, está associado um tipo de terreno. Cada tipo de terreno tem características diferentes no que toca à propagação do fogo. Na grelha existem outros tipos de elementos para além dos terrenos, nomeadamente os fogos, o bombeiro, o *bulldozer*, etc.

Figura 1. Exemplo de uma grelha de jogo. Na imagem podem-se ver terrenos de floresta de pinheiro (em cima à esquerda) e de eucalipto (em baixo à esquerda). Existe fogo em dois terrenos. Há partes do terreno de mato rasteiro (cantos superior e inferior direito) e terrenos de terra sem vegetação. Estão também presentes um bombeiro e um *bulldozer*.

(Imagens processadas com base em originais publicados em: https://www.pinterest.com/; https://www.shutterstock.com/ e https://opengameart.org/).

Descrição do jogo e implementação

Leitura dos elementos de jogo

No início do jogo são carregados os elementos de jogo a partir de um ficheiro de texto. No exemplo adjacente pode observar o ficheiro de configuração inicial que corresponde ao exemplo da figura 1. O ficheiro é composto por duas partes:

```
pppppmmmmm
pppp___mmm
ppp____mmp
ee_______p
e_________
eee_______
eeeee_____
eeeemm____
eeeeemm___
eeeeemmmmm
Fireman 5 3
Bulldozer 5 5
Fire 1 2
Fire 1 3
```

- Mapa dos terrenos – contém uma representação do mapa, onde cada linha do ficheiro corresponde a uma linha do mapa de jogo. Cada linha tem tantos caracteres quanto o número de colunas do mapa. Os caracteres utilizados para representar cada tipo de terreno são:
  - 'p' – pinheiro;
  - 'e' – eucalipto;
  - 'm' – mato rasteiro;
  - '_' – sem vegetação.
- Restantes elementos – após o mapa dos terrenos, segue-se um conjunto de linhas, onde cada uma delas representa um dos restantes elementos do jogo (*e.g.*, bombeiro, fogo, etc.) e a sua posição inicial na área de jogo.

Note que poderão existir vários mapas.

Requisitos do funcionamento do jogo

Após a leitura do ficheiro com o mapa e restantes elementos, o jogo deverá ter o seguinte funcionamento:

- O bombeiro é controlado pelo jogador através de teclas direcionais (as "setas" do teclado). Cada jogada corresponde a carregar numa tecla. O bombeiro pode avançar para qualquer posição da grelha exceto no caso de haver fogo nessa posição. Caso o jogador mova o bombeiro para cima do fogo, este apaga o fogo (visualizando-se um jato de água durante essa jogada).
- O fogo começa na posição ou posições indicadas no ficheiro. Em cada jogada, o fogo poderá propagar-se a casas vizinhas, com as seguintes probabilidades: 10% se a casa vizinha for eucalipto, 5% se for pinheiro e 15% se for mato rasteiro. Para efeitos de vizinhança consideram-se as posições da esquerda, direita, cima e baixo.
- Cada terreno que pegue fogo e não seja apagado é consumido ao fim de um certo número de jogadas: 10 no caso do pinheiro, 5 no caso do eucalipto e 3 no caso de mato rasteiro. A terra sem vegetação e os restantes elementos não são inflamáveis. Quando um terreno é consumido pelo fogo, deverá aparecer na posição correspondente uma imagem que corresponda ao terreno queimado.
- O *bulldozer* só se movimenta caso o bombeiro esteja dentro da máquina. Quando o bombeiro vai para a mesma posição do *bulldozer*, "entra" dentro do mesmo (nessa altura o bombeiro desaparece da imagem e o utilizador passa a controlar o bulldozer). Quando se carrega em 'ENTER', o bombeiro sai do *bulldozer*, fica na mesma célula deste, e passa novamente a ser controlado pelo utilizador. O *bulldozer* transforma todas as células por onde passa em terra sem vegetação, servindo assim para criar barreiras anti-fogo.

- Quando se carrega na tecla 'P', o bombeiro chama um avião. Este aparece inicialmente na parte inferior do mapa, na coluna onde existir uma maior quantidade de fogos. Após aparecer, o avião avança duas casas para cima em cada jogada e apaga os fogos das células por onde passar.
- Não é suposto o bombeiro ou o bulldozer pegarem fogo nem é suposto estes conseguirem avançar para posições da grelha que contém fogo. Já o avião pode passar por qualquer posição da grelha.
- Considera-se que um mapa está terminado quando já não existe fogo em nenhuma posição. Quando se termina um mapa, devem ser registadas as pontuações num ficheiro associado a esse e depois ser carregado automaticamente o próximo mapa. O ficheiro das pontuações deverá manter as 5 pontuações mais altas para o mapa em questão e os *nicknames* dos jogadores que as realizaram.
- O sistema de pontuações do jogo fica ao seu critério – deverá ser um sistema de pontuações que faça sentido dado o contexto (por exemplo, quando um fogo é apagado pelo bombeiro, recebem-se pontos, quando um fogo consome um terreno perdem-se pontos). Durante o jogo, a pontuação deverá aparecer na barra de estado (ver próxima secção).

Interface com o utilizador

Neste projeto, a interface com o utilizador é composta pela classe `ImageMatrixGUI` e pelo interface `ImageTile`, que estão incluídos no pacote `pt.iul.ista.poo.gui`. do projeto `GraphPack` fornecido.

A classe `ImageMatrixGUI` permite abrir uma janela como a representada na Figura 1. A área de jogo na janela pode ser vista como uma grelha bidimensional de 10x10 posições onde se podem desenhar imagens de 50x50 pixels em cada posição. Além da área de jogo, poderá existir uma barra de estado onde se podem mostrar informações sobre o jogo, nomeadamente as pontuações.

As imagens que são usadas para representar os elementos de jogo encontram-se na pasta "imagens", dentro do projeto. Para se desenhar a imagem de um objeto, este deverá implementar `ImageTile`:

```
public interface ImageTile {
	String getName();		// nome da imagem
	Point2D getPosition();	// posicao de desenho
	int getLayer();			// camada de desenho
}
```

As classes de objetos que implementarem `ImageTile` terão que indicar o nome da imagem a usar (sem extensão), a posição da grelha de jogo onde é para desenhar, e a camada de desenho (*layer*). Esta última determina a ordem pela qual as imagens são desenhadas, nos casos em que há várias sobrepostas na mesma posição (quanto maior o *layer*, "mais em cima" será desenhada a imagem).

Na classe `ImageMatrixGUI` estão disponíveis os seguintes métodos:

- `public void addImages(final List<ImageTile>)` – associa uma lista de objetos `ImageTile` à janela de desenho;
- `public void addImage(final ImageTile)` – associa um objeto `ImageTile` à janela de desenho;
- `public void removeImage(final ImageTile)` – remove um `ImageTile` da área de desenho;
- `public void clearImages()` – remove todos os `ImageTile` da área de desenho;
- `public void update()` – redesenhar os objetos ImageTile associados à janela de desenho;
- `public void setStatusMessage(final String message)` – modifica a mensagem que aparece na barra de estado.

O código fornecido inclui também o enumerado `Direction` e as classes `Point2D` e `Vector2D` (pacote `pt.iul.ista.poo.utils`). `Direction` representa as direções (UP, DOWN, LEFT, RIGHT) e inclui métodos úteis relacionados com as teclas direcionais do teclado. A classe `Point2D`, representa pontos no espaço 2D e deverá ser utilizada para representar os pontos da grelha de jogo. Inclui métodos para obter os pontos vizinhos e para obter o resultado da soma de um ponto com um vetor. Finalmente, a classe `Vector2D` representa vetores no espaço 2D.

A utilização dos pacotes fornecidos será explicada com maior detalhe nas aulas práticas.

Poderão vir a ser publicadas atualizações aos pacotes fornecidos caso sejam detetados *bugs* ou caso se introduzam funcionalidades adicionais que façam sentido no contexto do jogo.

É possível utilizar outras imagens, diferentes das fornecidas, para representar os elementos do jogo, ou para criar elementos de jogo adicionais. Não são esperados problemas desde que as imagens tenham uma dimensão de 50x50 pixels.

Estruturação do código

Sugere-se que o projeto siga um diagrama de classes baseado no que se apresenta na Fig. 2.

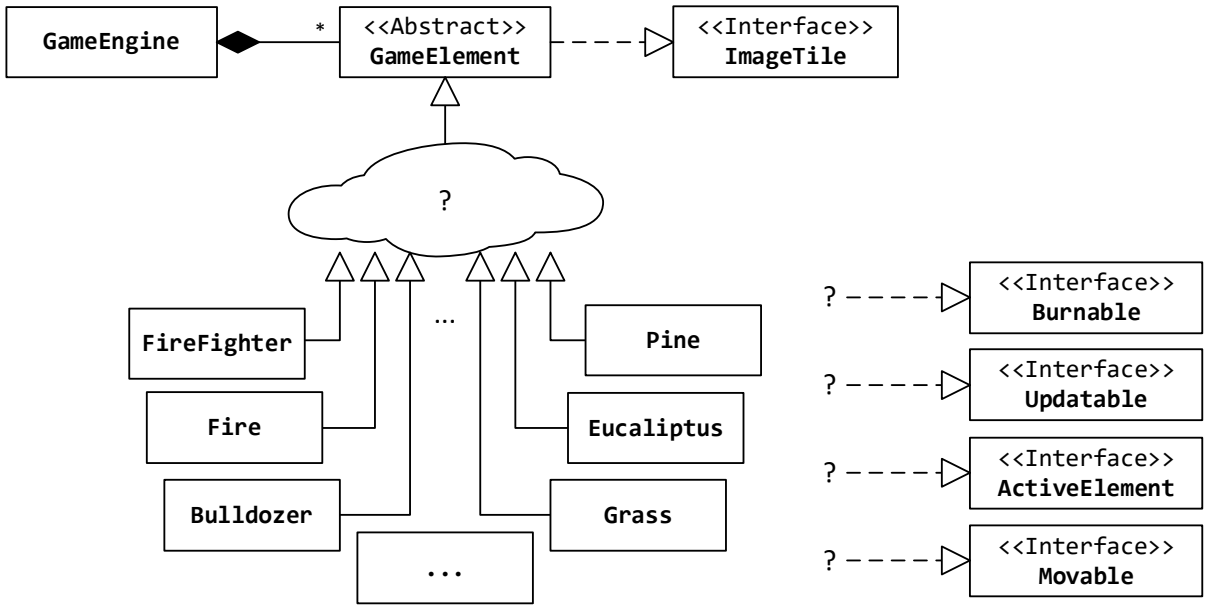

Figura 2 Esquema básico do diagrama de classes.

Existe uma classe central (`GameEngine`) que é responsável pela inicialização do jogo, manter as listas de objetos que correspondem aos elementos de jogo e despoletar cada jogada. Sugere-se que a classe central siga o padrão *solitão* (*singleton*) a fim de se facilitar a comunicação com as restantes classes.

Quanto aos elementos de jogo, estes devem ser hierarquizados de forma a tirar o máximo partido da herança, sendo derivados direta ou indiretamente de uma classe abstrata `GameElement`. Devem também ser definidas características dos elementos de jogo que possam ser modelizadas utilizando interfaces. Desta forma, os métodos que definir nos interfaces poderão ser invocados sem que seja necessário saber em concreto qual o tipo específico de objetos. Na Fig. 2 apresentam-se algumas possibilidades de títulos para interfaces no contexto do jogo, mas são apenas sugestões sem detalhes – fica a seu cargo definir interfaces que façam sentido e de tirar partido das mesmas.

Desenvolvimento do Projeto

Regras gerais

O trabalho deve ser feito por grupos de dois alunos e espera-se que em geral demore 30 a 40 horas a desenvolver. Poderá também ser feito individualmente (nesse caso o tempo de resolução poderá ser um pouco maior). Recomenda-se que os grupos sejam constituídos por estudantes com um nível de conhecimentos semelhante, para que ambos participem de forma equilibrada na execução do projeto e para que possam discutir entre si as opções de implementação.

É encorajada a partilha de ideias sobre o trabalho, **mas é inaceitável a partilha de código**. Caso sejam detetados trechos do mesmo código em trabalhos diferentes, os estudantes envolvidos ficam sujeitos ao que está previsto no código de conduta académica do ISCTE-IUL, arriscando-se a uma reprovação imediata a POO e a que o caso seja reportado à Reitora.

Acompanhamento do projeto

Contacte regularmente o docente das aulas práticas para que este vá revendo o projeto consigo, quer durante as aulas, quer nos horários de dúvidas (com marcação). Desse modo:

- evita opções erradas na fase inicial do projeto (opções essas que em geral conduzem a grandes perdas de tempo e escrita de muito mais código do que o necessário);
- a discussão final do trabalho poderá ser dispensada, dado que tanto os estudantes como o docente foram discutindo as opções tomadas.

Faseamento do projeto

Nesta secção apresenta-se um possível faseamento do projeto, baseado num esquema de *Action Points* semanais:

- Semana 1 (8 a 14/nov):
  - Entender a mecânica de comunicação entre o motor de jogo e a GUI;
  - Modelizar uma primeira aproximação à hierarquia de classes do jogo;
  - Ler o ficheiro de configuração e representar os elementos na GUI;
  - Implementar o movimento do bombeiro;
- Semana 2 (15 a 21/nov):
  - Implementar o modelo de propagação do fogo;
  - Implementar a interação bombeiro-fogo;
  - Ponderar ajustes ao diagrama de classes;
- Semana 3 (22 a 28/nov):
  - Implementar interação bombeiro-bulldozer;
  - Implementar movimento e interações do bulldozer;
  - Implementar o avião e suas interações;
  - Ponderar ajustes ao diagrama de classes;
- Semana 4 e início da semana 5 (29/nov a 8/dez)
  - Implementar a mudança de mapa;
  - Implementar o registo de pontuações em ficheiro;
  - Implementar os novos elementos de jogo (a divulgar mais tarde);
  - Rever/refinar opções tomadas, tendo em vista a tornar o programa mais flexível face à introdução de novos elementos de jogo.

A conclusão dos *Action Points* dentro dos tempos indicados não conta para avaliação mas, se houver muitos em atraso, significa que o projeto não está a ser desenvolvido ao ritmo que devia.

Avaliação

Objetivo

Realizar o projeto deverá ajudar a consolidar e a explorar os conceitos próprios de POO. Por isso, para além da componente funcional do trabalho, **é fundamental demonstrar a utilização correta da matéria própria de POO**, em particular:

- modularização do código e exploração das relações entre classes;
- utilização de herança e sobreposição de métodos com vista a evitar replicação de código;
- definição, implementação e utilização de interfaces, com vista a flexibilizar o código;

Mais à frente será publicada uma nova versão do enunciado onde se irão acrescentar alguns elementos ao jogo, com o intuito de testar se o seu desenvolvimento foi feito de modo a acomodar facilmente o crescimento do projeto. Poderá nessa altura reavaliar algumas opções e rever algum código com vista a tornar o seu projeto mais flexível, mais fácil de crescer, e com melhor nota!

Entrega

O prazo para entrega dos trabalhos é o **final do dia 8/dez de 2021**. A entrega é feita através da plataforma de *e-learning*.

Deverá cumprir escrupulosamente as seguintes regras de entrega:

- O material a entregar consiste num *archive file* gerado pelo eclipse que contém a exportação do seu projeto – **apenas o projeto que tem o jogo**, sem o `GraphPack`. Para exportar o projeto, dentro do eclipse deverá fazer *File/Export/Archive File/*, selecionar o projeto que tem o jogo, e exportar para um ficheiro zip.
- Deverá entregar esse zip no *e-learning*, na pasta que corresponde ao docente que acompanhou o seu trabalho ou, no caso do seu trabalho não ter sido acompanhado, na pasta dos projetos sem acompanhamento.
- Tenha em atenção que:
  - O nome do seu projeto (e não o zip do "export"), precisa de ser único e de identificar os elementos do grupo – por exemplo, `FireFight_TomasBrandao741_MariaAlbuquerque696` seria um nome válido. Para mudar o nome do projeto no eclipse faça *File/Refactor/Rename*.
  - O código do seu projeto não pode usar caracteres especiais (á, à é, ç, etc.).
  - Se possível, teste a importação do seu projeto num outro computador, antes de o entregar.

Os trabalhos que não cumprirem as regras em cima não se conseguem importar com facilidade para o eclipse, obrigando a realizar um *setup* manual do projeto, atrasando de forma muito significativa o processo de correção. Como tal, os **trabalhos mal entregues só serão tratados, corrigidos e discutidos após os autores dos mesmos estarem aprovados na parte escrita** (e, caso não fiquem aprovados, os trabalhos nem sequer serão vistos).

Critérios de avaliação

O trabalho será classificado com "A", "B", "C" ou "D", de acordo com os seguintes critérios:

- Grau de cumprimento dos requisitos funcionais;
- Utilização correta de herança;
- Definição e utilização correta de interfaces;

- Boas práticas no encapsulamento;
- Modularização, distribuição e legibilidade do código;
- Originalidade e extras (implementação de padrões, uso de comparadores, expressões lambda, etc.).

Serão imediatamente classificados com "D" os projetos que:

- contenham erros de sintaxe;
- não tenham um mínimo de requisitos funcionais implementados, nomeadamente o modelo de propagação do fogo, o movimento do bombeiro e a interação entre o bombeiro e fogo;
- tenham uma estrutura muito desadequada à programação orientada para objetos.

Note que, mesmo cumprindo estes mínimos, também poderão acabar com "D" os projetos que não demonstrem saber usar/aplicar os conceitos próprios de POO (herança, interfaces, etc.).

Por outro lado, para um projeto poder vir a ser classificado com "A", terá que cumprir todos os requisitos funcionais (ainda que com pequenos bugs que não sejam evidentes).

Discussão

Os projetos estão sujeitos a uma discussão, embora muitos estudantes possam vir a ser dispensados da mesma, caso os seus projetos tenham sido bem acompanhados pelo docente das aulas práticas. **As discussões são realizadas de 13 a 15/dez/2021**, durante os horários das aulas práticas.

A discussão é individual e os estudantes terão que demonstrar ser capazes de realizar um trabalho com qualidade igual ou superior ao que foi entregue. Os membros do grupo são implicitamente responsáveis pelo código do projeto entregue. Assume-se que os membros do grupo participaram de forma equilibrada na sua execução, tendo adquirido os conhecimentos necessários para produzir um trabalho do mesmo tipo individualmente. Caso um dos membros do grupo não consiga demonstrar esta capacidade na discussão poderá ficar com uma nota mais baixa do que a do projeto, ou até mesmo reprovar na UC.

Deteção de plágio e política em caso de fraude

Todos os projetos entregues serão submetidos a uma ferramenta antiplágio que analisa o código desenvolvido e devolve um índice de similaridade entre projetos. Nos casos em que houver indícios de fraude académica serão seguidos os procedimentos previstos no código de conduta académica do ISCTE-IUL.