Projeto de POO 2023/24 (v1.0)

SOKOBAN

Introdução

Neste projeto pretende-se criar um jogo conhecido como *Sokoban*[1], utilizando os principais conceitos de POO lecionados ao longo do semestre.

As principais características de um jogo deste tipo são:

- É um jogo de estratégia *turn-based* (uma jogada por cada ciclo de jogo).
- A personagem controlada pelo utilizador – neste caso a empilhadora – faz um movimento em cada jogada. O objetivo do jogo é arrumar um conjunto de caixotes em locais pré-definidos (marcados com "X"), empurrando-os e evitando que fiquem em posições de onde não é possível retirá-los. O jogador só consegue empurrar um caixote de cada vez.
- Ao terminar a arrumação num nível do jogo, passa-se para um novo nível que tipicamente tem um problema mais difícil. São valorizadas as soluções com o menor número de movimentos da empilhadora.

Uma ilustração básica do jogo a desenvolver pode ser observada na figura 1.

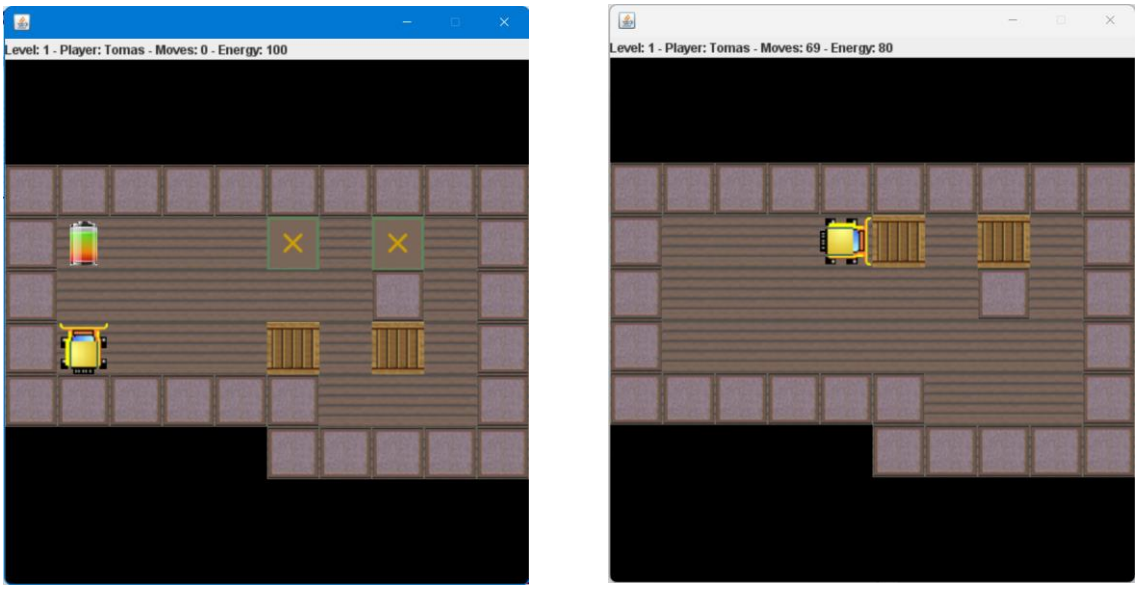

Fig. 1 - Exemplo do interface gráfico do jogo. Esquerda: estado inicial de um nível do jogo; Direita: conclusão do nível com a arrumação dos caixotes nos locais indicados – repare também que a bateria foi consumida para a empilhadora poder ter mais energia para se movimentar e empurrar caixotes.

No caso específico do projeto de POO, o jogo irá incluir elementos adicionais que fazem uma extensão ao jogo básico (teleportes, paredes quebráveis, paletes, etc.), cuja descrição poderá encontrar mais à frente neste enunciado.

Como o jogo é *turn-based*, cada jogada inicia-se cada vez que o utilizador prime uma tecla direcional. A empilhadora deverá mover-se segundo direção da tecla que tiver sido pressionada e interagir com eventuais objetos que ocupem a posição de destino: por exemplo empurrar um caixote na direção do

[1] https://en.wikipedia.org/wiki/Sokoban

movimento ou “apanhar” uma bateria.

A interação entre a empilhadora e os restantes objetos do jogo deve ser programada de uma forma flexível e tão autónoma quanto possível, distribuindo-se o código pelas classes correspondentes e minimizando-se a quantidade de código presente na classe que representa o motor de jogo.

No final de cada nível deve ser atribuída uma pontuação ao jogador, baseada no número de movimentos (quanto menos movimentos, melhor). Deve ser mantido um ou mais ficheiros de pontuações onde conste um Top-3 das melhores pontuações para cada nível, associadas aos nomes dos jogadores que as obtiveram. É possível perder o jogo no decurso de um nível, por exemplo se a empilhadora ficar sem energia, e nesse caso deve-se reiniciar o nível.

Leitura dos elementos de jogo

Cada nível é descrito por um ficheiro com o nome “`level`*N*`.txt`” em que *N* é o número do nível. O jogo deverá começar sempre no nível 0 (“`level0.txt`”). O estado inicial de cada nível do jogo é determinado por um ficheiro de configuração de “armazém”, com o formato ilustrado na Fig. 2.

```
=######===
##   X ####
#    T    #
#   XC    #
#          #
#          #
#     B    #
#          #
# TC     E#
##########
```

Fig. 2 – Exemplo de um ficheiro de configuração da sala inicial e correspondente representação gráfica.

O ficheiro contém a “planta” do armazém. Assume-se que todos os armazéns têm um tamanho fixo de 10 x 10 células. Cada símbolo representa um tipo de elemento de jogo – no exemplo da Fig. 2:

- os cardinais (‘#’) representam as paredes;
- os espaços em branco (‘ ’) representam o chão;
- os iguais (‘=’) representam espaço vazio;
- os ‘C’s representam caixotes;
- os ‘X’s representam os alvos, i.e., os pontos de arrumação dos caixotes;
- o ‘E’ é a empilhadora;
- os ‘B’s são baterias;
- os ‘T’s identificam portais de teleporte – só pode haver um par por nível;

Note também que a coordenada (0,0) corresponde ao canto superior esquerdo (a primeira linha corresponde a y=0, a segunda linha a y=1, e assim por diante).

Para os restantes elementos, os respetivos símbolos são indicados na próxima secção.

Características e comportamento dos elementos de jogo

Os elementos básicos do jogo, suas características e símbolos usados no ficheiro de configuração dos níveis, são os seguintes:

- **Empilhadora 'E' –** controlada pelo jogador; começa o jogo com 100 pontos de energia; de cada vez que se move desconta um ponto à energia, mas se empurrar algum objeto perde um ponto adicional. Caso a energia da empilhadora chegue a 0, perde-se o jogo.
- **Caixote 'C' –** pode ser empurrado pela empilhadora desde que o movimento do mesmo não seja obstruído por outro elemento do jogo que seja "sólido" (parede, outro caixote, bateria, etc.).
- **Alvo 'X' –** ponto de arrumação dos caixotes; se todos os alvos estiverem cobertos por caixotes, o nível fica concluído e muda-se para o próximo.
- **Bateria 'B' –** "carrega" a empilhadora com mais 50 pontos de energia; a empilhadora apanha a bateria ao deslocar-se para cima dela.
- **Parede '#' –** serve de delimitador da área de jogo – não é transponível.
- **Chão ' ' e Vazio '=' –** são elementos meramente estéticos, sem interações com outros elementos.

Adicionalmente aos elementos básicos do jogo *Sokoban*, neste projeto existem também os seguintes:

- **Buraco 'O' –** se a empilhadora cair num buraco, perde o jogo; se um caixote cair no buraco deve ser removido do jogo e, se não sobrarem caixotes suficientes cobrir os alvos, perde o jogo. Os buracos podem ser tapados por paletes.
- **Palete 'P' –** pode ser empurrada pela empilhadora usando as mesmas regras que se usam para empurrar caixotes; caso se empurre uma palete para o buraco, este fica tapado e passa a comportar-se como o chão.
- **Martelo 'M' –** pode ser apanhado pela empilhadora e a partir desse momento a empilhadora passa a poder "partir" paredes rachadas.
- **ParedeRachada '%'** – pode ser removida pela empilhadora, desde que esta tenha apanhado o martelo; depois de removida uma parede rachada, passa a estar chão no seu lugar.
- **Teleporte 'T'** – os "portais" de teleporte formam um par sempre que estiverem presentes num nível. Quando um objeto (qualquer) entra num portal de teleporte passa de imediato para a posição do outro portal do par. Caso o portal de destino tenha algo em cima, não acontece o teleporte.

Interface gráfico

Neste projeto, o interface gráfico está implementado através da classe `ImageMatrixGUI` e do interface `ImageTile`, que estão incluídos no pacote `pt.iscte.poo.gui`. do projeto `GraphPack` fornecido.

A classe `ImageMatrixGUI` permite abrir uma janela como a representada na Fig. . A área de jogo na janela pode ser vista como uma grelha 2D de 10x10 posições, onde se podem desenhar imagens de 50x50 pixels em cada posição. Além da área de jogo, deverão ser mostradas informações por cima da área de jogo (por exemplo, o nível em que está, a energia dsiponível, etc.).

As imagens que são usadas para representar os elementos de jogo encontram-se na pasta "images", dentro do projeto. Para se poder desenhar a imagem de um elemento do jogo, este deverá implementar o interface `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 a 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 mais do que um elemento na mesma posição (quanto maior o *layer*, "mais em cima" será desenhado o objeto).

Na classe `ImageMatrixGUI` estão disponíveis os seguintes métodos:

- `public void update()` – redesenhar os objetos ImageTile associados à janela de desenho – deve ser invocado no final de cada jogada, para atualizar a representação dos elementos de jogo.
- `public void addImages(final List<ImageTile>)` – envia uma lista de objetos `ImageTile` para a janela de desenho. Note que não é necessário voltar a enviar objetos `ImageTile` quando há alterações nos seus atributos – isso apenas contribuiria para tornar o jogo mais lento.
- `public void addImage(final ImageTile)` – envia 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 setStatusMessage(final String message)` – modifica a mensagem que aparece por cima da área de jogo.

O código fornecido inclui também o enumerado `Direction` e as classes `Point2D` e `Vector2D` (pacote `pt.iscte.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 (entre outros) 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. Nesse caso aconselha-se apenas a que as imagens alternativas tenham também uma dimensão de 50x50 pixels.

Estruturação do código

Sugere-se que comece por seguir um diagrama de classes baseado no que se apresenta na Fig. 3.

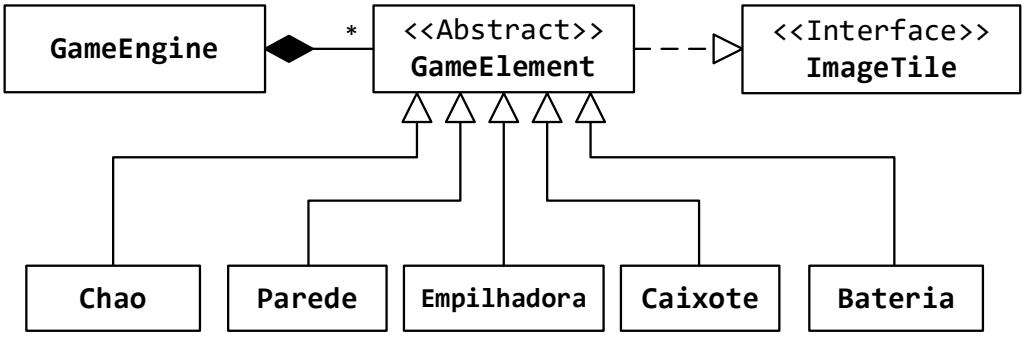

Fig. 3 – Desenho genérico, em UML, de algumas das principais classes e interfaces.

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 *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`. Note que poderá ser adequado utilizar vários níveis na hierarquia da herança.

Devem também ser definidas **características dos elementos de jogo que possam ser modelizadas utilizando interfaces**. Desta forma, os métodos declarados nos interfaces poderão ser invocados de uma forma mais abstrata, sem que seja necessário saber em concreto o tipo específico de objeto que o invoca. Fica a seu cargo definir interfaces que façam sentido e tirar partido dos mesmos.

Desenvolvimento e entrega do Projeto

Regras gerais

O projeto deve ser feito por grupos de dois alunos e espera-se que em geral demore 30 a 40 horas a desenvolver. Em casos justificados poderá ser feito individualmente (e 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, **mas é absolutamente inaceitável a partilha de código**. **Todos os projetos serão submetidos a software anti-plágio e, nos casos detetados, os estudantes envolvidos ficam sujeitos ao que está previsto no código de conduta académica do ISCTE-IUL, com reprovação a POO e abertura de um processo disciplinar.** Nos casos de partilha de código, os grupos envolvidos são penalizados da mesma forma, independentemente de serem os que copiam ou os que fornecem o código. Assume-se também que ambos os membros do grupo são responsáveis pelo projeto que entregaram.

Acompanhamento do projeto

Contacte regularmente o docente das aulas práticas para que este vá revendo o projeto consigo, quer durante as aulas, quer em sessões de dúvidas com marcação / por zoom. Desse modo evita opções erradas na fase inicial do projeto (opções essas que em geral conduzem a grandes perdas de tempo e a escrita de muito mais código do que o necessário).

Faseamento do projeto

Nesta secção apresenta-se o faseamento do projeto, baseado num esquema de *checkpoints*. Em cada uma das aulas práticas em que são verificados os checkpoints, as tarefas associadas já deverão estar concluídas e os estudantes já deverão estar a trabalhar nas tarefas do checkpoint seguinte.

- CheckPoint 1: aula prática entre 14 e 16/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 da empilhadora.
- CheckPoint 2: aula prática entre 21 e 23/nov
  - Implementar a interação da empilhadora com os caixotes;
  - Implementar a interação da empilhadora com a bateria;
  - Ponderar ajustes ao diagrama de classes;
- Checkpoint 3: aula prática entre 28 e 30/nov
  - Verificar a arrumação dos caixotes;
  - Implementar a mudança de nível;
  - Implementar o registo de pontuações em ficheiro;
- Checkpoint 4: aula prática entre 5 e 7/dez
  - Implementar os elementos adicionais (buraco, teleportes, paredes partidas, etc.);
  - Rever/refinar opções tomadas, tendo em vista a tornar o programa mais flexível face à introdução de novos elementos no jogo.

A conclusão dos checkpoints dentro dos tempos indicados conta para a componente de avaliação em aula – por isso não deixe que estes se atrasem!

Entrega do projeto

O prazo para entrega dos trabalhos é **23:59 de Sábado, dia 9/dez de 2023**. A entrega é feita através do *moodle*.

Para a entrega final do trabalho deve proceder da seguinte forma:

1. **Garantir que o nome do seu projeto, tanto no eclipse como no ficheiro zip, contém o nome e número de aluno de cada membro do grupo** – p.e., `Sokoban_TomasBrandao741_LuisNunes538` seria um nome válido. Para mudar o nome do projeto no eclipse faça *File/Refactor/Rename*.
2. **Gerar o *archive file* que contém a exportação do seu projeto – apenas o projeto onde 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.
3. **Entregar via *moodle*** o zip gerado no ponto anterior, na pasta de entrega de trabalhos a disponibilizar brevemente.
4. **Entregar um relatório**, cujo modelo será disponibilizado mais perto da data-limite para entrega.

Tenha também atenção ao seguinte:

- 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.

Caso não cumpra os procedimentos em cima, o seu projeto não se conseguirá importar com facilidade para o eclipse, obrigando os docentes a realizar *setups* manuais que atrasam de forma muito significativa o processo de avaliação dos projetos. Como tal, os **projetos mal entregues** (i.e., que não se consigam importar automaticamente para o eclipse a partir de um ficheiro zip) **serão penalizados com 2 valores.**

Avaliação

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 lecionada em POO**, em particular:

- Modularização e distribuição do código, explorando as relações entre classes;
- Utilização de herança e sobreposição de métodos com vista a evitar duplicação de código;
- Definição, implementação e utilização de interfaces, com vista a flexibilizar o código.

O trabalho será classificado de acordo com os seguintes critérios:

- Grau de cumprimento dos requisitos funcionais;
- Modularização, distribuição, encapsulamento e legibilidade do código;
- Definição e utilização de uma hierarquia adequada de herança;
- Definição e utilização correta de interfaces;
- Originalidade e extras (implementação de padrões, uso de comparadores, expressões lambda, tipos enumerados, coleções, etc.).

Serão imediatamente classificados com "0" (zero valores) os projetos que contenham erros de sintaxe ou que não implementem uma versão jogável num nível contendo os elementos básicos. Note que, mesmo cumprindo estes mínimos, poderão acabar com nota negativa os projetos que não demonstrem saber usar/aplicar os conceitos próprios de POO (herança, interfaces, etc.).

Discussão

O desempenho na discussão é avaliado individualmente. Na discussão, os estudantes terão que demonstrar ser capazes de realizar um trabalho com qualidade igual ou superior ao que foi entregue. Caso contrário ficam com notas mais baixas que a do trabalho, podendo inclusivamente reprovar caso tenham um mau desempenho na discussão.

Durante a discussão poderá ser solicitada a realização de pequenos trechos de código no âmbito do projeto ou serem pedidas alterações ao código existente.

As discussões serão realizadas de 11 a 15/dez/2023, preferencialmente durante os horários das aulas práticas.