Projeto de POO 2024/25 (v1.01)

Donkey-Kong

Introdução

O projeto consiste em criar um jogo similar ao *Donkey-Kong*[1]. As principais características deste tipo de jogo são as seguintes:

- É um jogo clássico de *arcade*, onde um personagem controlado pelo utilizador (*Jump-Man*) se move dentro de uma sala com várias plataformas e escadas;
- O adversário (*Donkey-Kong*) move-se de um modo independente do *Jump-Man*, lançando ataques do qual este tem de se desviar;
- Envolve a passagem do *Jump-Man* por várias de salas/níveis, desviando-se dos ataques do *Donkey-Kong* (bananas, armadilhas, etc) até chegar à última, onde se encontra o objetivo final do jogo (salvar a princesa);
- Os níveis vão ficando progressivamente mais difíceis, envolvendo um layout mais complexo e mais armadilhas.
- A "morte" do personagem implica voltar ao início.

Uma ilustração do jogo a desenvolver pode ser observada na figura 1 e pode ver um vídeo (na pasta do projeto) com um exemplo do desenrolar do jogo.

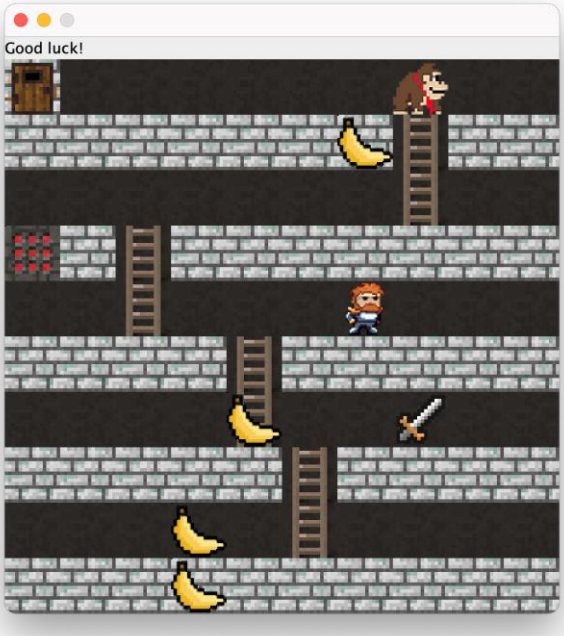

Fig. 1 - Exemplo do interface gráfico do jogo.

[1] https://pt.wikipedia.org/wiki/Donkey_Kong

No caso específico do projeto de POO, o jogo irá incluir elementos adicionais que fazem uma extensão ao jogo básico. **Podem ser dados a qualquer momento novos objetos ou adversários para implementar, quer durante a execução do trabalho, ou nas discussões, por isso a implementação flexível dessas classes é fundamental**.

Objetivos

Usando a interface gráfica fornecida pelos docentes em anexo ao enunciado, deve criar o "motor de jogo" para um jogo deste tipo. O motor de jogo deve permitir que o *Jump-Man* percorra diversas salas, enfrentando um adversário funcional e interagindo com objetos.

```
1   #2;room2.txt
2   WPG    G
3   WWWWWWWWWWS
4              S
5   SWWW  tt  WW
6   S  s
7   SWWWWW    WW
8   S     b
9   WWWWWWWWSWW
10             S
11  WWWWWWWWWWW
```

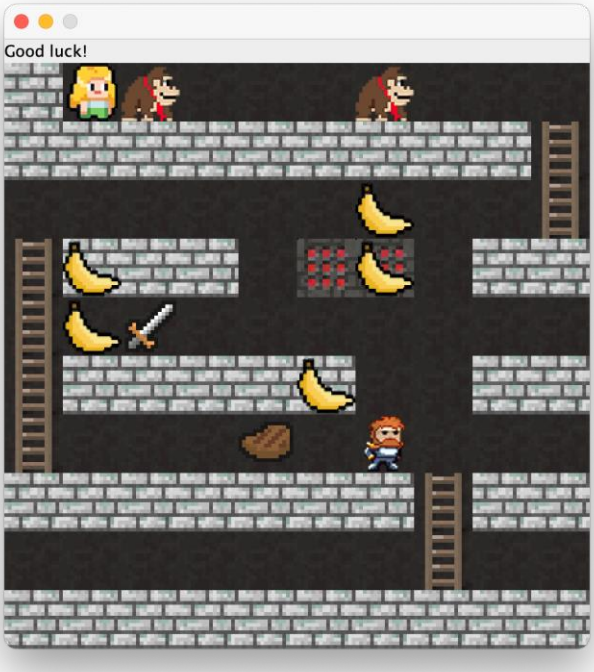

Fig. 2 – Exemplo de ficheiro da sala e da mesma sala durante o jogo (posições dos personagens móveis já não são as mesmas da configuração inicial). Na imagem da direita podemos ver, numa sala com uma porta, o Jump-Man ("herói" do nosso jogo) em baixo, o Donkey-Kong no topo a lançar bananas para o JumpMan, uma espada um bife e uma armadilha. O objetivo é passar pelo Donkey-Kong e chegar à princesa. As imagens usadas neste jogo são do jogo Pixel Dungeon (http://pixeldungeon.watabou.ru/) e encontram-se online em http://pixeldungeon.wikia.com/ (e são distribuídas com o exemplo inicial na pasta imagens).

Os objetos com os quais o *Jump-Man* pode interagir podem ter um impacto positivo, como reforçar a capacidade de ataque (por exemplo, uma espada), ou um impacto negativo, causando dano ao personagem (como armadilhas). O pedaço de carne ajuda a restabelecer a vida do personagem. Os objetos podem ser de interação única (desaparecem logo após o uso) ou de múltiplas interações (permanecem no jogo após a interação – como as armadilhas).

O *Jump-Man* não deverá atravessar certos objetos, como paredes ou adversários, e deverá "cair" caso não tenha nenhum suporte (não pode flutuar no ecrã). Mover-se para a mesma posição de um adversário (neste caso o *Donkey-Kong*) é considerado um ataque, e o adversário sofrerá um dano correspondente à capacidade de ataque do personagem. Da mesma forma, se o *Donkey-Kong* se mover para a posição do *Jump-Man*, este deverá sofrer o dano correspondente. Sempre que o *Jump-Man* apanhar um objeto ou sofrer um ataque de um adversário, a barra informativa deve exibir informações associadas à ação realizada.

O adversário movimenta-se aleatoriamente no início do jogo, mas nos níveis mais avançados aproxima-se progressivamente do *Jump-Man*. O adversário lança ataques constantes dos quais o *Jump-Man* se deve desviar até chegar ao adversário e alcançar a porta.

No final do jogo, deve ser apresentada ao jogador uma tabela de *highscore*s, calculada com base no tempo que o jogador levou até salvar a princesa. Os 10 melhores tempos (desde sempre) devem ser apresentados. Para isso, a informação deve ser armazenada de forma persistente no sistema de ficheiros.

Requisitos

Os mapas dos vários níveis (salas) devem ser lidos de um ficheiro com o formato indicado na secção "Objetivos". Para facilitar os testes, não se recomenda que a primeira sala tenha muitos objetos.

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 com o formato ilustrado na Fig. 2.

O motor de jogo recebe da interface gráfica informações sobre as teclas pressionadas. As teclas das setas devem controlar a movimentação do jogador.

O motor de jogo pode enviar imagens para a janela usando implementações de *ImageTile* (como as classes *JumpMan* e *Floor*, do exemplo fornecido). De cada vez que há uma alteração (por exemplo, mudança das posições das imagens) deve ser chamado o método update() do motor de jogo, de acordo com o padrão de desenho Observador/Observado.

A interface gráfica inclui também um *Ticker*, que é útil para gerir os movimentos automáticos do adversário. Dica: dentro da função *update()*, o motor de jogo deve verificar se houve um "tick" recente para diferenciar entre um movimento do Herói e a passagem do tempo, já que ambos acionam o método *update().* Isso permite que o jogo saiba quando o jogador realizou uma ação ou quando o tempo apenas avançou.

A interação entre objetos deve ser tão autónoma quanto possível (passando o mínimo possível pelo motor de jogo). O formato dos ficheiros onde são guardadas as melhores pontuações é livre.

A classe *ImageGUI* e a interface *ImageTile* são fornecidas, não devem ser alteradas e são fundamentais para a resolução do trabalho.

É obrigatória a utilização de:

- Herança de propriedades;
- Classe(s) abstrata(s);
- Implementação de interface(s);
- Estruturas de dados adequadas (e.g. Listas, Mapas...)
- Leitura e escrita de ficheiros;
- Exceções (lançamento e tratamento).

Interface gráfico

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

A classe `ImageGUI` permite abrir uma janela como a representada na Fig. 2. 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 disponível, 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 `ImageGUI` 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

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, **devendo ser derivados, direta ou indiretamente,** de uma classe-base abstrata. 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.

Contacte regularmente o docente das aulas práticas para que reveja 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 18 e 22/nov
  - Entender a mecânica de comunicação entre o motor de jogo e a GUI;
  - Criar modelo da hierarquia de classes do jogo (UML);
  - Ler o ficheiro de configuração e representar os elementos na GUI;
  - Implementar o movimento do *Jump-Man*.
- CheckPoint 2: aula prática entre 2 e 6/dez
  - Implementar a interação do *Jump-Man* com objetos (escada e um outro);
  - Implementar a movimento do *Donkey-Kong*;
  - Ponderar ajustes ao diagrama de classes;

- Checkpoint 3: aula prática entre 9 e 13-dez
  - Interação com outros objetos;
  - Implementar a mudança de nível;
  - Implementar elementos adicionais;
  - Rever/refinar opções tomadas, tendo em vista a tornar o programa mais flexível face à introdução de novos objetos / adversários 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 Domingo, dia 15/dez de 2024**. 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 nome do ficheiro zip, contém informação que vos permitam identificar: o nome e número-de-aluno de cada membro do grupo** – p.e., `DonkeyKong_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. **Se usou outro IDE envie o zip da pasta que tem todos os ficheiros do projeto, excluindo também a biblioteca GraphPack.**

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 alterações ao código nos pontos onde são expectáveis alterações.

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 17 a 20/dez/2024, preferencialmente durante os horários das aulas práticas. Dado o número de grupos é possível que sejam marcadas algumas discussões para fora do horário das aulas respetivas.