ISCTE-Instituto Universitário de Lisboa

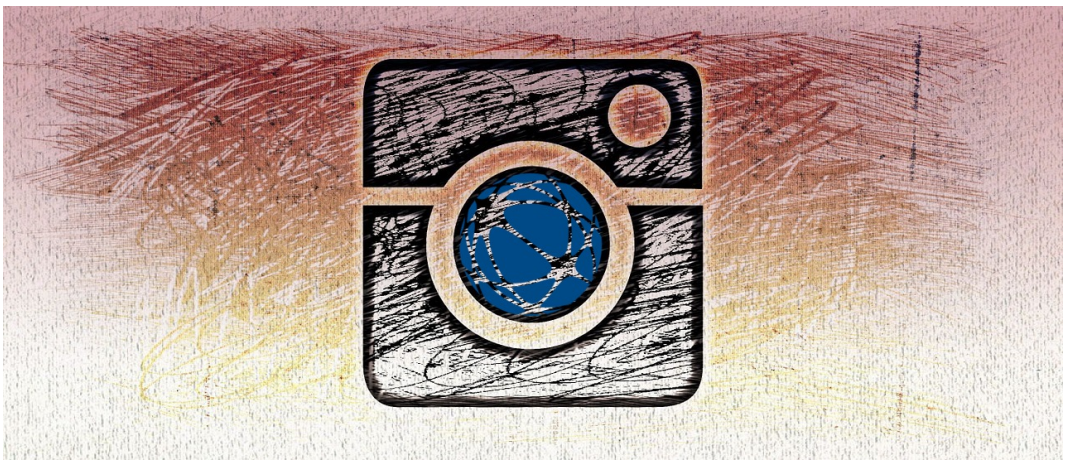
Introdução à Programação — 2017/2018 (1º semestre)

Projecto Individual

Data de entrega:	até à semana de 11 a 15 de Dezembro de 2017
Data da discussão:	até à semana de 18 a 22 de Dezembro de 2017

Introdução

Neste projeto pretende-se desenvolver o código que modela os principais conceitos de um editor de imagens (o ISCTeGRAM), onde será possível aplicar diversos efeitos numa imagem a cores existente. A solução desenvolvida deverá fazer uso das classes `ColorImage` e `Color` disponibilizadas nas aulas práticas. Este enunciado descreve todos os conceitos envolvidos.



Um editor de imagens permite editar uma imagem. A edição da imagem consiste na aplicação de diversos efeitos, designados **efeitos simples**. Um editor de imagens também deverá permitir a aplicação de efeitos constituídos por vários efeitos simples, designados **efeitos compostos**. Um editor de imagens deverá permitir desfazer a aplicação dos efeitos pela ordem inversa à da aplicação e refazer essa aplicação até que um novo efeito seja aplicado – **histórico** (*undo* e *redo*). Deverão ser desenvolvidas as seguintes classes de objetos:

- uma classe que define o conceito de editor de imagens;
- uma classe que define o conceito de efeitos composto.

Recomenda-se que o desenvolvimento do projeto decorra por etapas, as quais devem corresponder às secções que se apresentam em seguida. Não será apropriado avançar para uma etapa mais avançada sem ter a etapa anterior minimamente completa.

Editor de Imagens

Deverá ser desenvolvida uma classe para representar o editor de imagens. Para criar um objeto do tipo editor de imagens deverá ser passada uma imagem existente, da qual deverá ser feita uma cópia interna ao editor (de modo a não alterar o objeto da imagem original). Nesta classe existirão diversos métodos para manipular as imagens carregadas, nomeadamente através de:

- Efeitos simples;
- Efeitos compostos;
- Histórico (*undo* e *redo*).

Desenvolva todos os métodos e classes adicionais que achar necessário.

Efeitos Simples

Um efeito simples é um efeito que pode ser aplicado isoladamente. Os efeitos simples a desenvolver podem ter um inteiro como parâmetro ou não ter quaisquer parâmetros. Cada efeito deverá ser implementado num procedimento da classe principal. De seguida são apresentados os seis efeitos pretendidos: *noise*, *contrast*, *vignette*, *sepia*, *blur* e *film*.

Noise

O efeito de noise consiste em introduzir ruído na imagem. Deverá existir um parâmetro que permite definir a intensidade do ruído, sendo que quanto mais alto for o valor, maior será o ruído. O efeito pode ser implementado da seguinte forma:

para *cada píxel da imagem*
| sortear se o mesmo será alterado
| **se** *é para alterar* **então**
| | efetuar novo sorteio, de forma a determinar se o píxel será clareado ou escurecido
| | **se** *é para clarear* **então**
| | | alterar píxel para uma cor mais clara, de acordo com o parâmetro
| | **caso contrário**
| | | alterar píxel para uma cor mais escura, de acordo com o parâmetro
| | **fim**
| **fim**
fim

Figura 1: Exemplo de *noise*.

Contrast

O efeito de contrast consiste em realçar as diferenças de luminosidade dos píxeis da imagens, sendo que os píxeis mais escuros ficarão mais escuros, e os píxeis mais claros ficarão mais claros. Deverá haver um parâmetro que permite definir a intensidade do contraste.

Figura 2: Exemplo de *contrast*.

Vignette

O efeito de vignette consiste em escurecer os cantos da imagem de forma progressiva a partir de determinada distância do centro. Deverá existir um parâmetro que permite definir a partir de que distência do centro a imagem começa a escurecer. O efeito pode ser implementado da seguinte forma:

```
calcular o centro da imagem
para cada píxel da imagem
    calcular a sua distância ao centro
    se distância ao centro for maior que a distância mínima ao centro (parâmetro) então
        escurecer o píxel com uma intensidade relacionada com a distância (quanto mais
         distante, mais escuro)
    fim
fim
```

Figura 3: Exemplo de *vignette*.

Sepia

O efeito *sepia* não tem parâmetros e consiste em aplicar uma alteração à cor de todos os píxeis. A cor de cada píxel é substituída por uma nova cor, calculando os componentes RGB da seguinte

forma (sendo r, g e b, os valores originais):

$$R = 40\%r + 77\%g + 20\%b$$
$$G = 35\%r + 69\%g + 17\%b$$
$$B = 27\%r + 53\%g + 13\%b$$

Esta matriz de valores é um caso possível, porém outras variantes de pesos poderiam ser aplicadas. A implementação deste efeito deverá permitir uma fácil modificação destes valores.

Figura 4: Exemplo de *sepia*.

Blur

O efeito *blur* consiste em "esbater" os píxeis da imagem. Existem vários algoritmos para concretizar este efeito. Neste projeto pretende-se uma versão simples do mesmo que se explica de seguida.

A imagem alterada terá que ser construída de raiz, i.e. o efeito não poderá ser aplicado diretamente na imagem. O efeito deverá ter um parâmetro r que define a abrangência da vizinhança de cada píxel.

criar uma imagem com as dimensões da original
para *cada píxel da imagem nova*
  a cor é calculada com base nos píxeis vizinhos da imagem original (num raio r), sendo os valores RGB da nova cor calculados através de uma média entre os valores RGB das cores dos píxeis vizinhos
fim

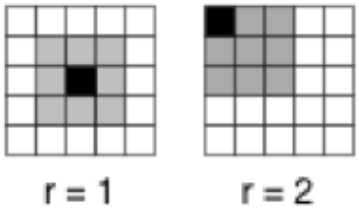


Figura 5: Como calcular a cor de um píxel através da média entre os valores RGB das cores dos píxeis vizinhos.

Figura 6: Exemplo de *blur*.

Film

O efeito *film* consiste em sobrepor na imagem algo do estilo apresentado na figura, simulando uma fita de negativo. A aparência da fita poderá não ser exatamente igual à apresentada, e poderá opcionalmente ser considerado um parâmetro para regular aspetos da fita (p.e. largura).

Figura 7: Exemplo de *film*.

Efeitos compostos

Considera-se um efeito composto como sendo um efeito definido em função de efeitos simples. Desta forma, um efeito composto consiste numa sequência de efeitos simples aplicados com valores predefinidos nos seus parâmetros (quando aplicável). Neste projeto pretende-se que seja possível definir efeitos compostos. Estes deverão ser representados numa classe, cujos objetos definem um efeito composto concreto que pode ser aplicado. A classe editor de imagens deverá ter um método que dado um objeto desta classe aplica o efeito composto (à semelhança dos efeitos simples). Haverá muitas formas de compor efeitos simples num efeito composto. As Figuras 8 e 9 apresentam dois exemplos de efeitos compostos.

Recomenda-se que a cada efeito simples seja associado um código. Estes códigos deverão estar definidos em constantes de forma a ser mais fácil a sua identificação e modificação. O seguinte excerto de código ilustra como definir constantes para os códigos.

```
public static final int NOISE = 0;
public static final int SEPIA = 1;
...
```

Esta classe deverá ter, pelo menos, um método que adiciona um efeito simples ao efeito composto representado por um dado objeto, de modo a que o seguinte código possa ser executado (tendo em conta as constantes definidas acima):

```
CompositeEffect oldPhoto = new CompositeEffect();
oldPhoto.add(SEPIA);
oldPhoto.add(NOISE, 10);
```

Figura 8: Efeito "retro", consistindo em aplicar *constrast*, *blur*, e *vignette*.

Figura 9: Efeito "old", consistindo em aplicar *noise*, *sepia*, e *vignette*.

Histórico

É comum encontrar em editores (não só de imagens) as funcionalidades de *undo* e *redo*. Estas funcionalidades baseiam-se num histórico de ações, de forma a poder anular ações efetuadas. *Undo* consiste em desfazer a última ação efetuada (neste projeto, a aplicação do último efeito, simples ou composto). Por exemplo, ao executar *undo* após aplicar *noise* na imagem original teríamos de novo a imagem original. *Redo* consiste em refazer a última ação desfeita. Por exemplo, ao executar *redo* após executar *undo* que anulou um efeito de *blur*, teríamos de novo a imagem com o *blur* aplicado anteriormente. Pretende-se que estas funcionalidades sejam implementadas no editor de imagens, devendo existir dois procedimentos para as mesmas.

Avaliação e Entrega

A realização do projeto é obrigatória para obter aprovação à Unidade Curricular (UC). **Não haverá qualquer possibilidade de obter aprovação à UC sem realizar o projeto.** A classificação no projeto não tem peso no cálculo da nota final. Contudo, a classificação no projeto define limites máximos para a mesma. O projeto será classificado da seguinte forma:

A, Muito bom (>80%):	a nota final obtida na UC poderá ser no máximo 20;
B, Bom (≤80%):	a nota final obtida na UC poderá ser no máximo 16;
C, Suficiente (≤60%):	a nota final obtida na UC poderá ser no máximo 12;
D, Não aprovado (<50%):	implica reprovação à UC.

O projeto será inicialmente avaliado em termos funcionais, i.e., se os objectos do tipo das classes a desenvolver têm o comportamento esperado, independentemente da forma como estão implementados, de acordo com os seguintes pesos: editor com efeitos simples, 50%; efeitos compostos, 25%; histórico, 25%. Desta primeira avaliação resultará uma classificação (A, B, C, ou D). Em função da qualidade do código poderá ser aplicada uma penalização que implica descer um nível na classificação, p.e.:

- Classificação funcional C com má qualidade de código, é despromovida para D;
- Classificação funcional A com má qualidade de código, é despromovida para B.

Os alunos poderão obter *feedback* juntos dos professores das respetivas turmas sobre o progresso do projeto e qualidade do código, e deverão seguir as recomendações dadas.

Os projetos só poderão ser entregues presencialmente em suporte eletrónico (ficheiros `.java`), diretamente ao professor da turma a que o aluno pertence.

Política em caso de fraude

Os alunos podem partilhar e/ou trocar ideias entre si, sobre os trabalhos e/ou resolução dos mesmos. No entanto, o trabalho entregue deve corresponder ao esforço individual de cada aluno. *O projeto é individual, e em nenhum caso deve ser copiado código que será entregue.* A deteção de código copiado será realizada por software especializado bastante sofisticado. Casos de plágio óbvio serão penalizados com a anulação do projeto, o que implica a reprovação à Unidade Curricular. Adicionalmente, a situação será reportada à Comissão Pedagógica da ISTA e ao Conselho Pedagógico do ISCTE-IUL. Serão penalizados da mesma forma tanto os alunos que fornecem código como os que copiam código de outros.