COLÉGIO DO SAGRADO CORAÇÃO DE MARIA LISBOA

APLICAÇÕES INFORMÁTICAS B

Projeto 3 – GIMP CSCM

Projeto 2.º Período Ano letivo 2015/2016

Aluno (a) _____ N.º_____

Data ___/ ___/ ____ Ano ___.º Turma _____

O Docente _____

Introdução

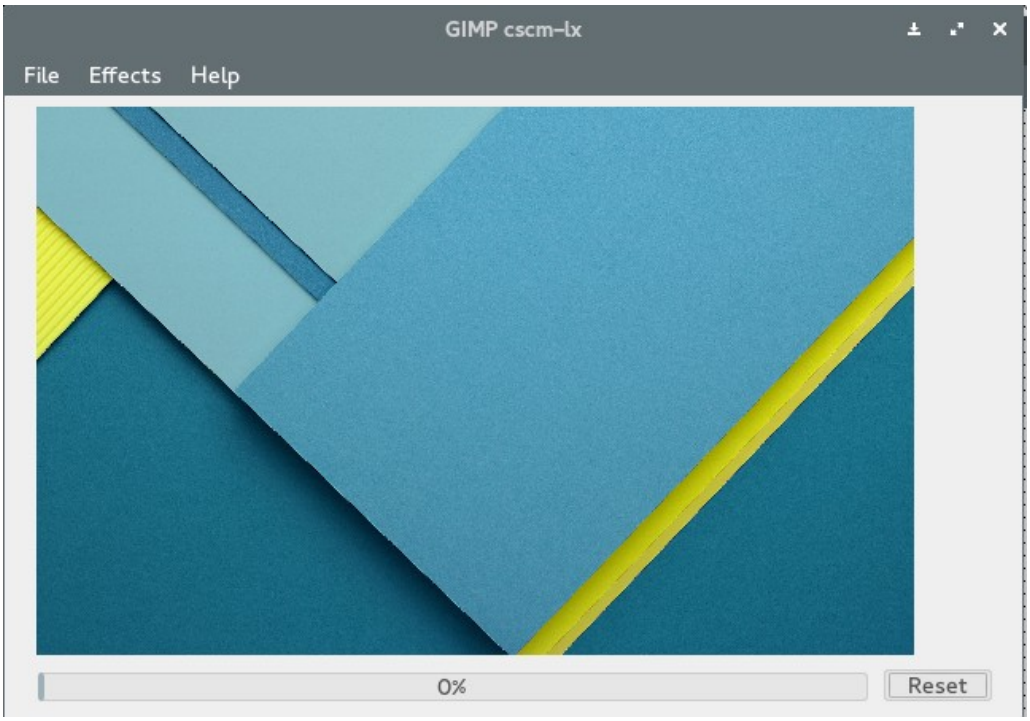
Este projeto tem como objetivo fundamental o de mobilizar os conhecimentos adquiridos na área da programação orientada a eventos de modo a construir um editor de imagem básico.

Condições e Duração

Este projeto, a pares (podendo haver exceções), tem a duração de 6 aulas de 60 minutos e deve ser realizado maioritariamente em sala de aula com a supervisão do docente.

Requisitos

Deverá seguir as seguintes indicações que contém os requisitos do projeto.



O programa deverá cumprir ainda os seguintes requisitos:

#1. O programa deverá ser configurado para suportar multilíngua; *[10 pontos]*

#2. O programa deverá contemplar, pelo menos os idiomas português e inglês (*pt* e *en*); *[10 pontos]*

#3. O *Form* principal deverá ter o título de "GIMP cscm-lx"; *[5 pontos]*

#4. Deverá conter um menu principal com os seguintes itens: *[55 pontos]*

i. File > Open – Deverá abrir uma nova imagem através do *TOpenDialog* e carregá-la num objeto *Timage [10 pontos]*

ii. File > Save as BMP – Deverá guardar o ficheiro no formato BMP no disco *[20 pontos]*

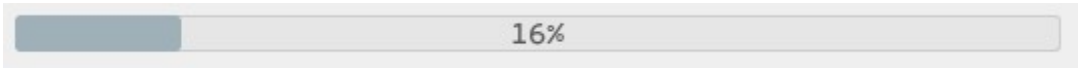
iii. File > Exit – deverá sair da aplicação *[5 pontos]*

```
Application.Terminate;
```

iv. Effects > Menu que vai executar cada um dos efeitos descritos mais à frente neste documento *[5 pontos – só o menu]*

v. Help > About – Deverá abrir um novo *form* que deverá conter: Logo (original e personalizado) , Versão do programa e Autores; *[15 pontos]*

#5. Deverá conter uma *TprogressBar [10 pontos]*



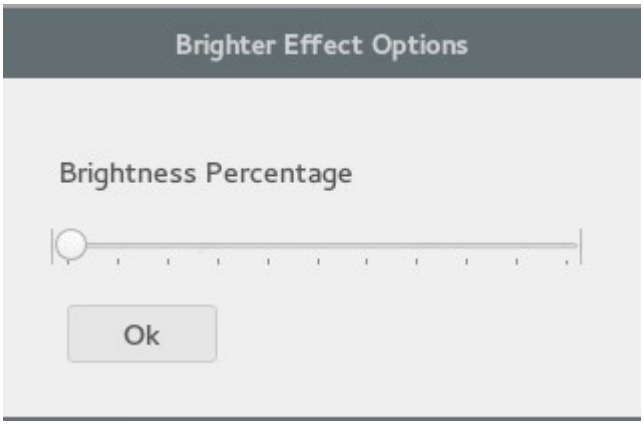
i. Deverá apresentar a percentagem de execução do efeito selecionado para a imagem

#6. Deverá conter um botão de *reset* que volta a carregar a imagem original [5 *pontos*]

#7. Efeitos a aplicar:

i. **Brighter Effect** – Torna uma imagem mais clara. Deverá abrir um novo *form* onde se indica a percentagem *[10 pontos]*

1. Para cada Pixel a quantidade de cada uma das cores RGB segue a fórmula:
    1. R := R + R*(Brightness percentage);
    2. G := G + G*(Brightness percentage);
    3. B := B + B*(Brightness percentage);
    4. onde *Brightness percentage* é o valor da Form2.TrackBar1.Position / 10;



ii. **RedChannel Effect** – Deixa ficar em cada pixel apenas a a quantidade de vermelho. A de verde e azul devem ficar a 0. *[10 pontos]*

iii. **GreenChannel Effect** – Deixa ficar em cada pixel apenas a a quantidade de verde. A de vermelho e azul devem ficar a 0.*[10 pontos]*

iv. **BlueChannel Effect** – Deixa ficar em cada pixel apenas a a quantidade de azul. A de vermelho e verde devem ficar a 0.*[10 pontos]*

v. **Gray Scale** – Deve transformar uma imagem a cores em tons de cinzento:*[10 pontos]*

1. //Luminance Y := 0.2126 R + 0.7152 G + 0.0722 B

2. //Gray = RGB(Y, Y, Y)

vi. **Negative** – Deve transformar uma imagem no seu negativo, para isso os novos valores de RGB de cada píxel devem ser iguais a (255 - valor atual) – o complementar *[10 pontos]*

vii. **Box Blur** – Aplicar um filtro de desfocagem

1. O valor de RGB de cada Pixel é igual à média do valor dos píxeis adjacentes e dele próprio.*[20 pontos]*

Avaliação

O seguintes itens serão considerados na avaliação: *[25 pontos]*

- O código fonte: *[18 pontos]*
  - Deverá estar devidamente documentado com comentários pertinentes; *[6 pontos]*
  - Deverá estar identado e formatado segundo as regras de estilo que estão a ser utilizadas nas aulas e que podem ser verificadas no portfólio do professor; *[6 pontos]*
  - Deverá estar estruturado o mais possível em sub-rotinas (*functions* e *procedures*); Cada *function* ou *procedure* deverá ter um comentário onde se descreve o seu objetivo. *[6 pontos]*
- O programa deve compilar sem erros e sem avisos (*warnings*); *[7 pontos]*
- O programa deverá correr sem erros de lógica cumprindo integralmente todas as funcionalidades pedidas nos requisitos técnicos;
- No final os alunos poderão ser chamados a “defender” os eu trabalho, podendo a classificação final ser alterada consoante o desempenho do aluno.

Algumas dicas

Abrir um TOpenDialog:

```
if OpenDialog1.Execute then
 begin
   FileName := OpenDialog1.Filename; //Caminho completo para o ficheiro selecionado
 end;
```

Carregar uma imagem num TImage a partir de um caminho:

```
Form1.Image1.Picture.LoadFromFile(FileName);
```

Passar por todos os pixeis de uma TImage:

```
var i, j: Integer;
  clr: TColor;
  r,g,b: Byte;
(...)
 for i := 0 to Image1.Picture.Width do
  begin
      for j := 0 to Image1.Picture.Height do
      begin
        clr := Image1.Picture.Bitmap.Canvas.Pixels[i,j];
        r := Red(clr); //
        g := Green(clr);
        b := Blue(clr);
```

```
      end;
      Application.ProcessMessages; //Deixar o programa processar outras mensagens
  end;
```

Alterar o valor de um pixel:

```
 Image1.Picture.Bitmap.Canvas.Pixels[i,j] := RGBToColor(10, 20, 30);
```

Alterar o valor de uma progress bar:

Utilizar a propriedade Position que pode receber um inteiro entre o e 100.

```
ProgressBar1.Position := 50; //Coloca a 50%
```

Obter o valor de uma TrackBar que está no form2:

```
  Form2.ShowModal(); //Mostra o form2
  position := Form2.TrackBar1.Position; //obter a posição da trackbar (0..10)
```

Fechar um form através de um botão:

```
  Form2.Close;
```

Trabalhar numa cópia da imagem original:

```
var
  Bitmap: Tbitmap;
begin

 //Criar um bitmap com o tamanho da imagem
  Bitmap := TBitmap.Create;
  Bitmap.Height := Image1.Picture.Bitmap.Height;
  Bitmap.Width := Image1.Picture.Bitmap.Width;

  //Alterar os pixeis do Bitmap
  //(...)
  Bitmap.Canvas.Pixels[i, j] := RGBToColor(10, 20, 30);
  //(...)

  //Substituir o Bitmap da TImage pelo Bitmap que criámos
  Image1.Picture.Bitmap.Canvas.Draw(0, 0, Bitmap);

  //Libertar a memória reservada para o Bitmap
  Bitmap.Free;

end;
```

Trabalhar com ficheiros binários

Declarar uma variável para o ficheiro

```
var f: file;
```

Abrir o ficheiro para escrita

```
assignFile(f, saveFileName);
rewrite(f,1);
```

Escrever no ficheiro a variável w:

```
blockwrite(f,w,sizeof(w));
```

Fechar o ficheiro

```
closeFile(f);
```

Tipo de Variável	N.º Bytes
Byte	1
Word	2
Integer	4

Estrutura Ficheiro BMP (Simplificado)

File Header (14 bytes)

Image Header (40 bytes)

Pixel Data (Depende do n.º de píxeis)

1-File Header (14 bytes)

Campo	Tamanho (bytes)	Valor	Descrição
bfType	1	'B'	Fixo 'B'
bfType	1	'M'	Fixo 'M'
BfSize (file size)	4 (integer)		Tamanho calculado do ficheiro em bytes
bfReserved1	2 (word)	byte(0)	Fixo 0
bfReserved2	2 (word)	byte(0)	Fixo 0
bfOffBits	4 (integer)	integer(54)	Fixo 54 Posição onde começa o bloco Pixel Data

2-Image Header (40 bytes)

Campo	Tamanho (bytes)	Valor	Descrição
biSize	4 (integer)	integer(40)	Fixo 40 Tamanho em bytes do Image Header
biWidth	4 (integer)	integer(width)	Largura da imagem em pixeis
biHeight	4 (integer)	integer(height)	Altura da imagem em pixeis
biPlanes	2 (word)	word(1)	Fixo de 1
biBitCount	2 (word)	word(24)	Fixo 24 Bits per pixel
biCompression	4 (integer)	integer(0)	Fixo 0

biSizeImage	4 (integer)	integer(0)	Fixo 0
biXPelsPerMeter	4 (integer)	integer(0)	Fixo 0
biYPelsPerMeter	4 (integer)	integer(0)	Fixo 0
biClrUsed	4 (integer)	integer(0)	Fixo 0
biClrImportant	4 (integer)	integer(0)	Fixo 0

3-Pixel Data

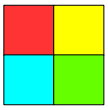
Linha a linha de baixo para cima e da esquerda para a direita

Os valores escritos no ficheiro são pela seguinte ordem: BGR (Blue, Green, Red)

Cada linha tem de ter um número de bytes múltiplo de 4. Preencher os restantes com o valor 0.

Exemplo 1:

Considere a seguinte imagem com uma resolução de (2x2) → 2 pixeis de largura e 2 pixeis de altura



***Pixel Data*:**

BGR (Blue Green Red) → *bottom to top, left to right*

255 255 0

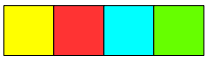
0 255 0 → 1.ª linha - faltam 2 bytes para a linha ficar múltipla de 4 (acrescentar **0 0**)

0 0 255

0 255 255 → 2.ª linha - faltam 2 bytes para a linha ficar múltipla de 4 (acrescentar **0 0**)

Exemplo 2:

Considere a seguinte imagem com uma resolução de (1x4) → 1 pixel de largura e 2 pixeis de altura



***Pixel Data*:**

BGR (Blue Green Red) → *bottom to top, left to right*

0 255 255

0 0 255

255 255 0

0 255 0→ Neste caso o número de bytes já é múltiplo de 4

Bom Trabalho!!