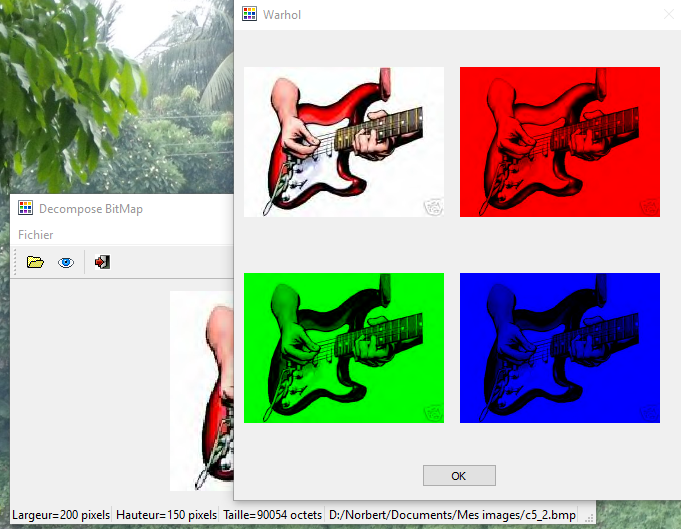
**Algorithme principal (correspond à on\_actionOuvrir\_Triggered)**

|  |
| --- |
|  |
| Mini Projet 2 |
|  |
| **Décomposer un BMP** |

Ce programme a pour objectif de décomposer une image au **format BMP de profondeur 24** en 3 images ne comportant chacune que les couleurs élémentaires Rouge, Vert et Bleu.



Contraintes imposées

* L’utilisateur doit, avec une fenêtre de dialogue, pouvoir choisir l’image originale.
* Le programme reposera essentiellement sur la manipulation de tableaux dynamiques pour l’extraction. Des bibliothèques spécifiques pour traiter ce problèmes sont proscrites (QColor par exemple).
* Le programme ne traitera que les images de profondeur 24 (si besoin refaire une image avec paint.net par exemple).

**nomFichierBMP :** chaine de caractère

**tailleBMP, offsetBMP** : entierlong

**DEBUT**

Saisir ***nomFichierBMP*** avec la fen^tre de dialogue appropriée

SI nomFichierBMP est valide

ALORS

Lire l’entête du fichier BMP et restituer tailleBMP et offsetBMP

Initialiser un tableau dynamique par couleur

Annuler les couleurs bleu et vert du tableau rouge

Annuler les couleurs rouge et vert du tableau bleu

Annuler les couleurs rouge et bleu du tableau vert

Sauver les tableaux dans des fichiers

FINSI

FIN

# Algorithme extraire rouge

DEBUT

POUR i s’incrémentant par pas de 3 de offsetBMP JUSQU'A tailleBMP

FAIRE

tableauRouge[i]=0

tableauRouge[i+1]=0

FINPOUR

FIN

Programme complet

# Mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QLabel>

#include <QtWidgets>

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~MainWindow();

void getTexte(QString &largeur,QString &hauteur, QString &taille);

public slots:

void on\_actionQuitter\_triggered();

void on\_actionOuvrir\_triggered();

void on\_actionVoir\_triggered();

private:

Ui::MainWindow \*ui;

unsigned char enteteBmp[54];

unsigned char \* tableauRouge;

unsigned char \* tableauVert;

unsigned char \* tableauBleu;

QLabel \*largeurBMP;

QLabel \*hauteurBMP;

QLabel \*tailleBMP;

QLabel \*imageLbl;

QLabel \*nomFichierBMP;

int largeur, hauteur, taille, offset;

void ouvrirFichier(QString fichier);

void afficherDonneesBmp(QString nom);

int rendValeur(int debut);

void writeSizes(int l,int h, int t);

void afficherBmpOriginal(QString nomFichier);

void copierTableau(unsigned char \*tableau);

void extraireRouge();

void extraireBleu();

void extraireVert();

void sauverCouleurs();

};

#endif // MAINWINDOW\_H

# dialog.h

#ifndef DIALOG\_H

#define DIALOG\_H

#include <QDialog>

#include "mainwindow.h"

namespace Ui {

class Dialog;

}

class Dialog : public QDialog

{

Q\_OBJECT

public:

explicit Dialog(QWidget \*parent = 0);

~*Dialog*();

void getName(QString nom);

private:

Ui::Dialog \*ui;

QLabel \*orgBmp;

QLabel \*redBmp;

QLabel \*greenBmp;

QLabel \*blueBmp;

};

#endif // DIALOG\_H

# Mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "dialog.h"

#include "ui\_dialog.h"

#include <QtWidgets/QFileDialog>

#include <QDebug>

#include <math.h>

//----------------------------------------------------------------

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

//initialise la barre d'état

largeurBMP=new QLabel();

hauteurBMP=new QLabel();

tailleBMP=new QLabel();

nomFichierBMP=new QLabel();

ui->statusBar->addWidget(largeurBMP,1);

ui->statusBar->addWidget(hauteurBMP,2);

ui->statusBar->addWidget(tailleBMP,3);

ui->statusBar->addWidget(nomFichierBMP,4);

imageLbl=new QLabel;

imageLbl->setAlignment(Qt::AlignCenter);

setCentralWidget(imageLbl);//incruste l'image dans le centralWidget

ui->actionVoir->setDisabled(true);//désactive le menu initialement

}

//-----------------------------------------------------------

MainWindow::~*MainWindow*()

{//destructeur de la fenêtre

delete ui;

}

//-----------------------------------------------------------

void MainWindow::on\_actionQuitter\_triggered()

{//pour quitter le programme alt+X est le raccourcis

close();

}

//------------------------------------------------------------

void MainWindow::on\_actionOuvrir\_triggered()

{ //ouverture du fichier

QString nomFichier = "";

nomFichier=QFileDialog::getOpenFileName(this, "Ouvrir une image", QString(), "Images (\*.bmp)");

if (!nomFichier.isEmpty())

{

ouvrirFichier(nomFichier);

afficherDonneesBmp(nomFichier);

afficherBmpOriginal(nomFichier);

extraireBleu();

extraireVert();

extraireRouge();

sauverCouleurs();

}

}

//-------------------------------------------------------------

void MainWindow::ouvrirFichier(QString fichier)

{//Ouvre le fichier et place l'entête dans le tableau d'entête puis copie

//le contenu dans les tableaux de couleur

QFile fichierBmp(fichier);

if (fichierBmp.*open*(QIODevice::ReadOnly))

{//le fichier est lisible

//on lit l'entête

fichierBmp.read((char \*)enteteBmp,sizeof(enteteBmp));

//on récupère taille, largeur, hauteur et offset

taille=rendValeur(2);

largeur=rendValeur(18);

hauteur=rendValeur(22);

offset=rendValeur(10);

//on place le fichier dans chaque tableau de couleur

tableauRouge= new unsigned char[taille];

tableauVert= new unsigned char[taille];

tableauBleu= new unsigned char[taille];

fichierBmp.*seek*(0);//on retourne au début du fichier

fichierBmp.read((char \*)tableauRouge, taille);

fichierBmp.*seek*(0);//on retourne au début du fichier

fichierBmp.read((char \*)tableauVert, taille);

fichierBmp.*seek*(0);//on retourne au début du fichier

fichierBmp.read((char \*)tableauBleu, taille);

fichierBmp.*close*();

}

else

QMessageBox::critical(this,"Erreur","Le fichier "+fichier+"/n ne peut être ouvert.");

}

//---------------------------------------------------------------------

void MainWindow::afficherDonneesBmp(QString nom)

{ //on affiche les données dans la barre d'état

nomFichierBMP->setText(nom);

writeSizes(largeur, hauteur, taille);

}

//---------------------------------------------------------------

void MainWindow::afficherBmpOriginal(QString nomFichier)

{ //affiche le bmp sur MainWindow

QPixmap \*pix = new QPixmap(nomFichier);

imageLbl->setPixmap(pix->scaled(300,200,Qt::KeepAspectRatio));

}

//------------------------------------------------------------------

int MainWindow::rendValeur(int debut)

{ //rend en le nombre correspondant commencant à début et terminant à debut+4

long temp=0;

for (int i=debut;i<debut+4;i++)

temp=temp+enteteBmp[i]\*pow(256,(i-debut));

return temp;

}

//---------------------------------------------------------------------

void MainWindow::writeSizes(int l,int h, int t)

{ //affiche les valeurs dans la barre d'états

QString temp;

temp="Largeur="+QString::number(l)+" pixels";

largeurBMP->setText(temp);

temp="Hauteur="+QString::number(h)+" pixels";

hauteurBMP->setText(temp);

temp="Taille="+QString::number(t)+" octets";

tailleBMP->setText(temp);

}

//---------------------------------------------------------------------

void MainWindow::extraireBleu()

{//

for (int i=offset;i<taille;i+=3)

{

tableauBleu[i+1]=0;

tableauBleu[i+2]=0;

}

}

//---------------------------------------------------------------------

void MainWindow::extraireRouge()

{//

for (int i=offset;i<taille;i+=3)

{

tableauRouge[i]=0;

tableauRouge[i+1]=0;

}

}

//----------------------------------------------------------------------

void MainWindow::extraireVert()

{//

for (int i=offset;i<taille;i+=3)

{

tableauVert[i]=0;

tableauVert[i+2]=0;

}

}

//----------------------------------------------------------------------

void MainWindow::sauverCouleurs()

{//on sauve les couches de couleurs dans des fichiers

QFile fichier("bleu.bmp");

if (fichier.*open*(QIODevice::ReadWrite))

{

fichier.write((char\*)tableauBleu,taille);

fichier.*close*();

}

fichier.setFileName("rouge.bmp");

if (fichier.*open*(QIODevice::ReadWrite))

{

fichier.write((char\*)tableauRouge,taille);

fichier.*close*();

}

fichier.setFileName("vert.bmp");

if (fichier.*open*(QIODevice::ReadWrite))

{

fichier.write((char\*)tableauVert,taille);

fichier.*close*();

}

ui->actionVoir->setDisabled(false);

}

//------------------------------------------------------

void MainWindow::on\_actionVoir\_triggered()

{

Dialog \*BmpDialog = new Dialog;

BmpDialog->setWindowModality(Qt::ApplicationModal);//on rend le dialog modal

BmpDialog->getName(nomFichierBMP->text());

BmpDialog->show();

}

# dialog.cpp

#include "dialog.h"

#include "ui\_dialog.h"

Dialog::Dialog(QWidget \*parent) :

QDialog(parent),

ui(new Ui::Dialog)

{

ui->setupUi(this);

setFixedSize(450,470);

setWindowFlags(Qt::CustomizeWindowHint);

setWindowFlags(Qt::WindowTitleHint);

setWindowFlags(Qt::WindowSystemMenuHint);

MainWindow m;

orgBmp = new QLabel;

redBmp = new QLabel;

greenBmp = new QLabel;

blueBmp = new QLabel;

ui->gridLayout->addWidget(orgBmp,0,0);

ui->gridLayout->addWidget(redBmp,0,1);

ui->gridLayout->addWidget(greenBmp,1,0);

ui->gridLayout->addWidget(blueBmp,1,1);

}

//---------------------------------------------------------

Dialog::~*Dialog*()

{

delete ui;

}

//-----------------------------------------------

void Dialog::getName(QString nom)

{ //affiche les données de tailles

QPixmap \*orgPix = new QPixmap(nom);

QPixmap \*redPix = new QPixmap("rouge.bmp");

QPixmap \*greenPix = new QPixmap("vert.bmp");

QPixmap \*bluePix = new QPixmap("bleu.bmp");

orgBmp->setPixmap(orgPix->scaled(200,200,Qt::KeepAspectRatio));

redBmp->setPixmap(redPix->scaled(200,200,Qt::KeepAspectRatio));

greenBmp->setPixmap(greenPix->scaled(200,200,Qt::KeepAspectRatio));

blueBmp->setPixmap(bluePix->scaled(200,200,Qt::KeepAspectRatio));

}

# main.cpp

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}