Haute Ecole de la Province de Liège : Département Informatique
==========

Graduat en Informatique et Systèmes : finalité Informatique Industrielle

LABORATOIRE D'INFORMATIQUE INDUSTRIELLE
==========

Traitement d'image laboratoire

3ème année

Dossier 1 : Etape 1 & 2
==========

MARTIN SIMON

Année Académique 2020 - 2021

Groupe : 2321

Table des matières

Description projet

Dans le cadre du cours de traitement d'image, il nous a été demandé de créer une application fenêtrée permettant d'afficher une image et de réaliser différentes modifications sur une image.

Pour créer celle-ci j'ai utilisé le langage java et j'utilise l'IDE NetBeans comme environnement de développement.

Choix et justification du langage

Pour réaliser ce projet, j'ai décidé d'utiliser le langage JAVA. Ce langage est très puissant car il permet la portabilité entre les différents OS alors que d'autres langages ne le permettent pas.

Java a d'autres avantages comme sa simplicité, sa robustesse et sa sécurité.

Au niveau de sa simplicité, Java n'utilise pas les pointeurs, la récupération de l'espace mémoire alloué dynamiquement (il est récupéré automatiquement), la surcharge des opérateurs, l'héritage multiple par rapport au C++.

Au niveau de sa robustesse, l'absence de pointeurs et l'obligation de définir sans ambiguïté les variables utilisées rend impossible des accès involontaires à des zones mémoire protégées ou inaccessibles.

Et pour finir au niveau de la sécurité, tout est prévu pour que le transfert d'une applet (unité de code téléchargeable) ou l'exécution d'une application ne puisse endommager le site d'exécution.

Java est un langage qui permet aussi de créer une application graphique bien plus facilement que le langage C ou C++ grâce au la librairie SWING et AWT.

Fonctionnement

Ouverture d'une image

```
private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {

    JFileChooser jfc = new JFileChooser(FileSystemView.getFileSystemView().getHomeDirectory());
    jfc.setDialogTitle("Select an image");
    jfc.setAcceptAllFileFilterUsed(false);
    jfc.setFileSelectionMode(JFileChooser.FILES_ONLY);
    FileNameExtensionFilter filter = new FileNameExtensionFilter("JPEG, BMP and GIF images", "jpg", "png", "bmp", "gif");
    jfc.addChoosableFileFilter(filter);

    int returnValue = jfc.showOpenDialog(null);
    if (returnValue == JFileChooser.APPROVE_OPTION) {
        System.out.println(jfc.getSelectedFile().getPath());
        path = jfc.getSelectedFile().getPath();

        try {
            imageFirst = convertToBuffered(ImageIO.read(new File(path)));
        } catch (IOException ex) {
            Logger.getLogger(windowMain.class.getName()).log(Level.SEVERE, null, ex);
        }

        setBufferedSource(imageFirst);
```

JFileChooser permet d'ouvrir une boite de dialogue permettant de naviguer dans les différents dossiers de l'ordinateur et sélectionner l'image que l'on veut.

FileNameExtensionFilter permet de spécifier quels formats d'image l'utilisateur doit sélectionner.

Pour finir, j'implémente la variable imageFirst qui de type BufferedImage avec la fonction ImageIO.read(new File(path)).

Zoom

```
private void ComboBoxZoomItemStateChanged(java.awt.event.ItemEvent evt) {

        int zoom = ComboBoxZoom.getSelectedIndex();

        Image tmpImage = imageSource.getSubimage(0, 0, imageSource.getWidth(), imageSource.getHeight());

        switch(zoom)
        {
            case 0: tmpImage = tmpImage.getScaledInstance(imageSource.getWidth()/2, imageSource.getHeight()/2, Image.SCALE_DEFAULT);
                break;

            case 1: tmpImage = tmpImage.getScaledInstance(imageSource.getWidth(), imageSource.getHeight(), Image.SCALE_DEFAULT);
                break;

            case 2: tmpImage = tmpImage.getScaledInstance(imageSource.getWidth()*2, imageSource.getHeight()*2, Image.SCALE_DEFAULT);
                break;

            case 3: tmpImage = tmpImage.getScaledInstance(imageSource.getWidth()*5, imageSource.getHeight()*5, Image.SCALE_DEFAULT);
                break;
        }

        BufferedImage tmp = convertToBuffered(tmpImage);
        setBufferedSource(tmp);
}
```

La fonction getScaledInstance() permet de redimensionner la taille de l'image en donnant en paramètre la largeur et la hauteur.

En fonction de la valeur de la combobox je vais multiplier celles-ci par la valeur sélectionnée par l'utilisateur.

Palette

```
private void setColor(Color newColor)
{
    int pixel;
    int r, g, b, a;
    int newR, newG, newB, newA;

    newR = newColor.getRed();
    newG = newColor.getGreen();
    newB = newColor.getBlue();
    newA = newColor.getAlpha();

    for(int i = 0; i < tmpBufferedImage.getWidth(); i++)
    {
        for(int j = 0; j < tmpBufferedImage.getHeight(); j++)
        {
            pixel = tmpBufferedImage.getRGB(i, j);

            a = (pixel >> 24)&0xff;
            r = (pixel >> 16)&0xff;
            g = (pixel >> 8)&0xff;
            b = pixel&0xff;

            a+=newA;
            if(a > 255)
                a = 255;
            if(a < 0)
                a = 0;

            r+=newR;
            if(r > 255)
                r = 255;
            if(r < 0)
                r = 0;

            g+=newG;
            if(g > 255)
                g = 255;
            if(g < 0)
                g = 0;

            b+=newB;
            if(b > 255)
                b = 255;
            if(b < 0)
                b = 0;

            pixel = (a<<24) | (r<<16) | (g<<8) | b;
            tmpBufferedImage.setRGB(i, j, pixel);
        }
    }
```

La fonction setColor() parcourt toute l'image et décompose chaque pixel en RGB ensuite elle rajoute à chacun d'eux la couleur sélectionnée.

ROI

```
private void Button_ROIActionPerformed(java.awt.event.ActionEvent evt) {
    if(tmpROI.getXRoi() == 0 || tmpROI.getYRoi() == 0 || tmpROI.getWidthRoi() == 0 || tmpROI.getHeightRoi() == 0)
    {
        JOptionPane.showMessageDialog(new JFrame(), "Selectionner une partie de l'image.", "Erreur", JOptionPane.ERROR_MESSAGE);
    }
    else {
        BufferedImage tmp = imageSource.getSubimage(tmpROI.getXRoi(), tmpROI.getYRoi(), tmpROI.getWidthRoi(),tmpROI.getHeightRoi());
        setBufferedDestination(tmp);
    }

}
```

Pour séletionner une zone d'intérêt on utilise la fonction getSubimage() dont en paramètre on passe les valeurs x et y et la largeur et la hauteur de la partie de l'image que l'on veut sélectionner.

Conclusion

Lors de la réalisation du projet j'ai rencontré certains soucis lors de la programmation de la fonction de la palette. J'ai finalement trouvé sur différents sites des explications pour programmer correctement cette fonction. Lors de la modification d'image j'ai pu constater que plus l'image était grande plus la modification prenait du temps à se faire. Cela s'explique car le programme analyse chaque pixel un à un et le modifie si nécessaire.