Cvičení 10 IPOG1_2019/2020

Téma

Práce s rastrovými obrázky a s jednotlivými pixely rastrového obrázku.

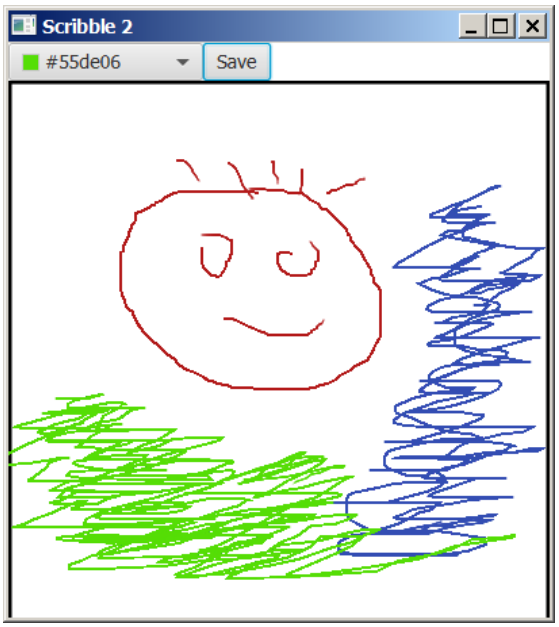
Cíl

- **Seznámit se s možnostmi načtení a zobrazení rastrových obrázků**
- **Seznámit se s možnostmi modifikace rastrových obrázků**
- **Seznámit se s přístupem k jednotlivým pixelům rastrového obrázku**
- **Vyzkoušet uložení vytvořeného nebo modifikovaného rastrového obrázku**

1. Uložení vytvořeného obrázku

Upravte aplikaci kreslení (Scribble) z minulého cvičení tak, aby bylo možno kreslit vybranou barvou, případně doplňte další atributy kreslení (šířka čáry, vzor, …).

Pro nastavení barvy kreslení použijte komponentu **`ColorPicker`** a při kreslení použijte nastavenou barvu získanou pomocí **`colorPicker.getValue()`**.



Doplňte možnost uložení nakresleného obrázku.

```
public void handle(ActionEvent t) {
    FileChooser fileChooser = new FileChooser();

    FileChooser.ExtensionFilter extFilter =
            new FileChooser.ExtensionFilter("png files (*.png)", "*.png");
    fileChooser.getExtensionFilters().add(extFilter);

    File file = fileChooser.showSaveDialog(primaryStage);

    if(file != null){
        try {
            WritableImage writableImage = new WritableImage(canvasWidth, canvaHeight);
            canvas.snapshot(null, writableImage);

            RenderedImage renderedImage = SwingFXUtils.fromFXImage(writableImage, null);
            ImageIO.write(renderedImage, "png", file);
        } catch (IOException ex) {
            System.out.println("File outpur error");
        }
    }
```

2. Jednoduché načtení a zobrazení obrázku

Vyzkoušejte práci s instancí třídy **`Image`** a **`ImageView`**.
Realizujte jednoduché načtení obrázku a jeho zobrazení.


```
Image image = new Image("beruska.jpg");
ImageView imageView = new ImageView();
imageView.setImage(image);


StackPane root = new StackPane();
root.getChildren().add(imageView);
Scene scene = new Scene(root, 300, 250);
primaryStage.setTitle("Image Read Test");
primaryStage.setScene(scene);
primaryStage.show();
```

3. Získání informací o obrázku

Zjistěte o obrázku základní informace (šířka a výška v pixelech).

```
System.out.println("Image Width: " + image.getWidth());
System.out.println("Image Height: " + image.getHeight());
```

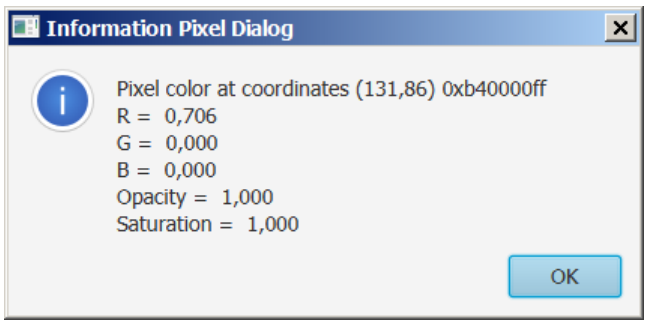
Získejte instanci třídy **`PixelReader`**.

```
PixelReader pixelReader = image.getPixelReader();
System.out.println("Pixel Format: " + pixelReader.getPixelFormat());
```

Vyzkoušejte přístup k jednotlivým pixelům pomocí instance třídy **`PixelReader`**.

```
Color color = pixelReader.getColor(x, y);
message = String.format("Pixel color at coordinates (%d,%d) %s\n", x,y,color);
message+=String.format("R =  %1.3f\n", color.getRed());
message+=String.format("G =  %1.3f\n", color.getGreen());
message+=String.format("B =  %1.3f\n", color.getBlue());
message+=String.format("Opacity =  %1.3f\n", color.getOpacity());
message+=String.format("Saturation =  %1.3f\n", color.getSaturation());
```

Zobrazte informaci o pixelu po stisknutí LT myši na ploše obrázku.



4. Zobrazení histogramu

Pomocí instance třídy **`PixelReader`** zjistěte četnost jasů pro jednotlivé barevné kanály a četnost celkového jasu jednotlivých pixelů pro načtený obrázek a získané četnosti zobrazte ve formě histogramu. Pro vizualizaci využijte vhodného grafu.

5. Dialog pro načtení vybraného obrázku

V předchozím příkladu (příkladech) doplňte možnost výběru požadovaného obrázku pomocí třídy **`FileChooser`**.

```
try {
    File file = fileChooser.showOpenDialog(stage);
    if (file!=null) {
        Image newImage = new Image(file.toURI().toString());

        if (newImage.isError() || (newImage==null)) {
            throw new Exception("Invalid picture file");
        }

        //...

    }
} catch (Exception e) {
    Alert alert = new Alert(Alert.AlertType.ERROR);
    alert.setHeaderText("Invalid picture format or invalid file.");
    alert.showAndWait();
}
```

6. Přizpůsobení zobrazení obrázku

Předchozí příklad pro jednoduché zobrazení obrázku doplňte o možnosti nastavení centrování, velikosti a zachování originálních poměrů stran.

Možnosti nastavení:

- Zobrazení v originální velikosti
- Zobrazení přes celou plochu rodičovské komponenty, bez zachování originálního poměru stran
- Zobrazení v maximální velikosti vzhledem k rozměrům rodičovské komponenty se zachováním originálního poměru stran
- Centrování v rámci rodičovské komponenty
- Otočení o vybraný úhel (případně překlopení/zrcadlení) s využitím transformací

7. Modifikace jednotlivých pixelů

Pomocí instance třídy **`WritableImage`** a **`PixelWriter`** můžete získat bitmapu, ve které můžete modifikovat barvu jednotlivých pixelů.

```
WritableImage wImage = new WritableImage(
        (int)image.getWidth(),
        (int)image.getHeight());
PixelWriter pixelWriter = wImage.getPixelWriter();
```

Pomocí třídy **`PixelReader`** (zmíněné v předchozím příkladu 3) můžete získat informaci o barvě pixelů v originálním obrázku a dle potřeby tuto barvu modifikovat a zapsat do nově vytvořené instance třídy **`WritableImage`**.

```
PixelReader pixelReader = image.getPixelReader();

for(int y=0;y<image.getHeight();y++){
    for(int x=0; x<image.getWidth();x++){
        Color color = pixelReader.getColor(x,y);
        color = color.brighter();
        pixelWriter.setColor(x,y,color);
    }
}
```

Pomocí tohoto přístupu proveďte vybrané barevné korekce (např. úprava jasu, kontrastu, odstranění vybraného barevného kanálu, separace vybraného barevného kanálu, …) načteného obrázku.