Język JAVA
podstawy programowania

Jacek Rumiński

Wykład 5, część 3

Plan wykładu:

1. Wprowadzenie do grafiki w Javie
2. Budowa GUI: komponenty, kontenery i układanie komponentów
3. Budowa GUI: obsługa zdarzeń
4. Grafika wektorowa – porysujmy sobie
5. Reprezentacja koloru
6. Grafika rastrowa - obrazy
7. Obsługa czcionek

Kolor

Standardowo przyjmuje się, że kolor na podstawie przestrzenni RGB definiowany jest jako liniowa kombinacja barw podstawowych:

$$kolor = r*R+g*G+b*B,$$

gdzie RBG to kolory podstawowe, a rgb to wartości wag z zakresu (0,1).

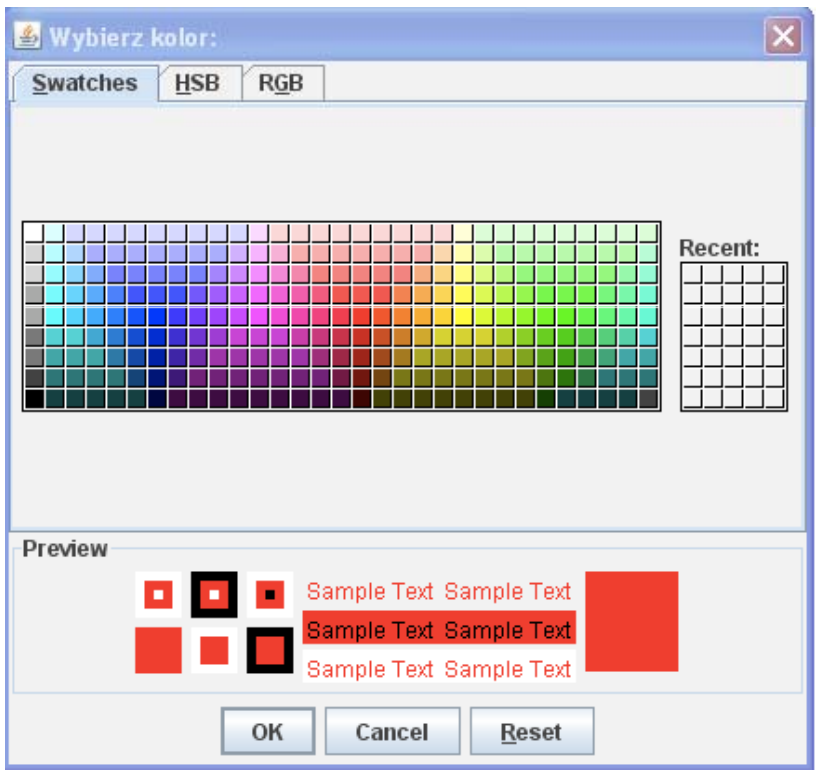
Zakres wag w systemach komputerowych jest przeważnie modyfikowany do postaci zakresu dyskretnych wartości całkowitych o dynamice 8 bitowej czyli 0..255. Stąd też programista definiuje kolor jako zbiór trzech wag z zakresu 0..255.

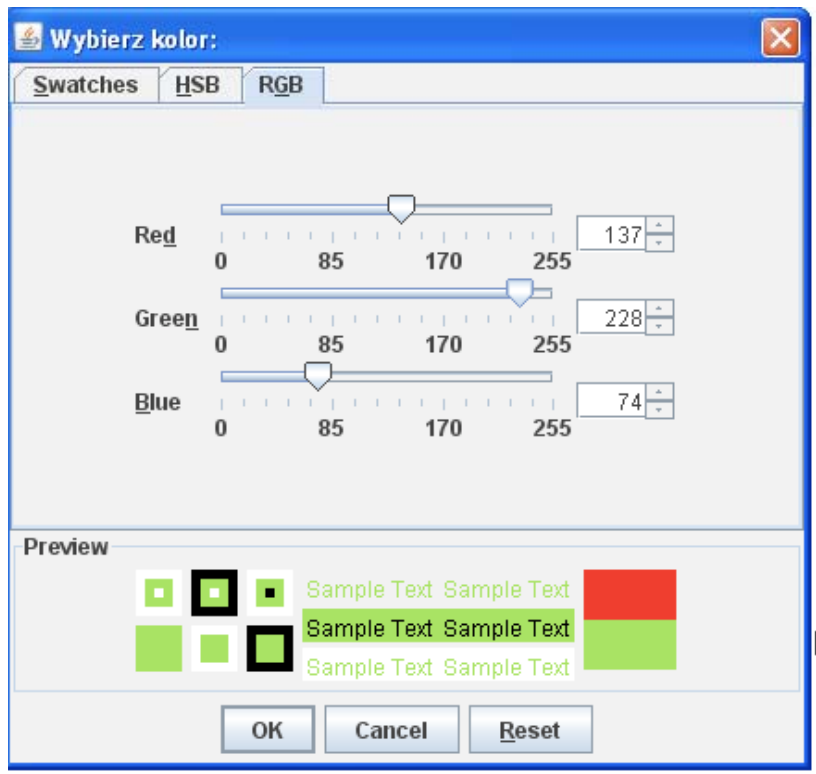
W Javie istnieje klasa Color, która bezpośrednio umożliwia nam zdefiniowanie koloru w modelu RGB, np. new Color(0,0,255) - daje barwę niebieską (R=0, G=0, B=255). W Javie istnieją predefiniowane stałe statyczne w klasie Color reprezentujące najczęściej wykorzystywane kolory np. Color.blue.

Kolor

Wartości wag dla koloru w systemie RGB możemy uzyskać stosując komponent Javy JColorChooser:

```
c = JColorChooser.showDialog(((Component)
e.getSource()).getParent(), "Wybierz kolor:", Color.red);
```





Język JAVA – kolor


```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class WyborKoloru extends JFrame {
 JButton jb_test;   Color c;
 public WyborKoloru() {    super("Wybierz kolor!");    c=Color.red;    initGUI();  }
 public void initGUI(){
    setSize(300, 300);   jb_test=new JButton("TEST");
   JButton jb_choose = new JButton("Wybierz kolor...");
   jb_choose.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {
       c = JColorChooser.showDialog(((Component) e.getSource()).getParent(), "K:", Color.red);
       jb_test.setBackground(c);       }    });
   add(jb_choose, BorderLayout.NORTH);    add(jb_test, BorderLayout.CENTER);
   setDefaultCloseOperation(EXIT_ON_CLOSE);
 }//koniec initGUI()
 public static void main(String args[]) {
   WyborKoloru wk = new WyborKoloru();
   wk.setVisible(true);   }//koniec main()
}//koniec class
```

Kolor

Kolor definiowany jest również w innych systemach kolorów, np. referencyjnym CIE XYZ, HSI, HSB, Lab, Luv, YUV, YCrCb, itd. Każdy z tych systemów ma swoje znaczenie. Przykładowo model typu YUV (Y-luminancja, U i V – składowe chrominancji) ułatwiają proces kompresji JPEG/MPEG poprzez większą kompresję stratną składowych chrominancji, na które oko ludzkie jest mniej wrażliwe.

Przykładowo kodowanie 4:2:2 oznacza, że na 4 piksele luminancji (w wierszu) przypadają dwa piksele chrominancji (np. 4Y 2U i 2V). W kodowaniu usuwanych jest połowa wartości pikseli chrominancji dla każdego komponentu! W kodowaniu 4:1:1 pozostaje tylko ¼ wartości chrominancji dla każdego komponentu.

Przy dekodowaniu stosuje się różne techniki (powielanie, filtracja) w celu eliminacji widocznych błędów (najczęściej efektu "schodkowego").

Na kolejnym slajdzie pokazano znaczenie takich kompresji.

Kolor

Luminancja (Y) co 4 wartość, U i V bez zmian

Chrominancja (U i V) co 4 wartość, Y bez zmian

ORYGINAŁ
(1100/830)

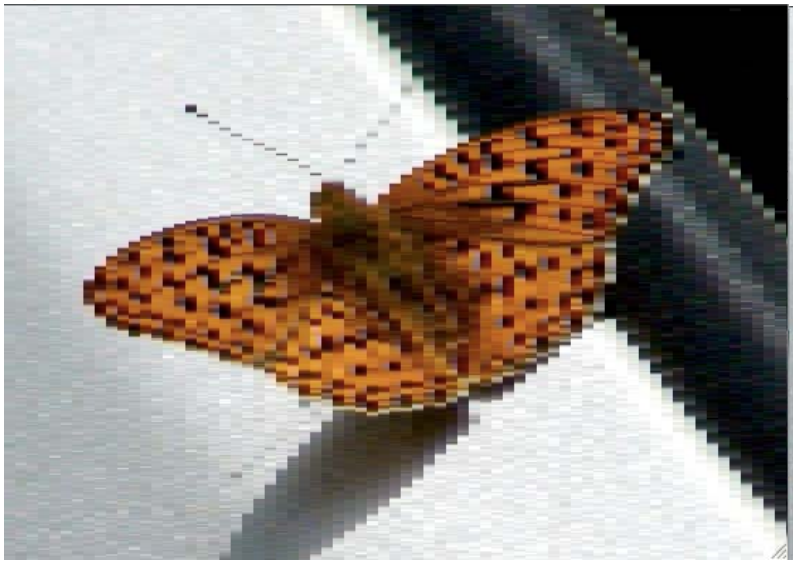
Kolor

Luminancja (Y) co 8 wartość, U i V bez zmian

Chrominancja (U i V) co 8 wartość, Y bez zmian

ORYGINAŁ

Kolor

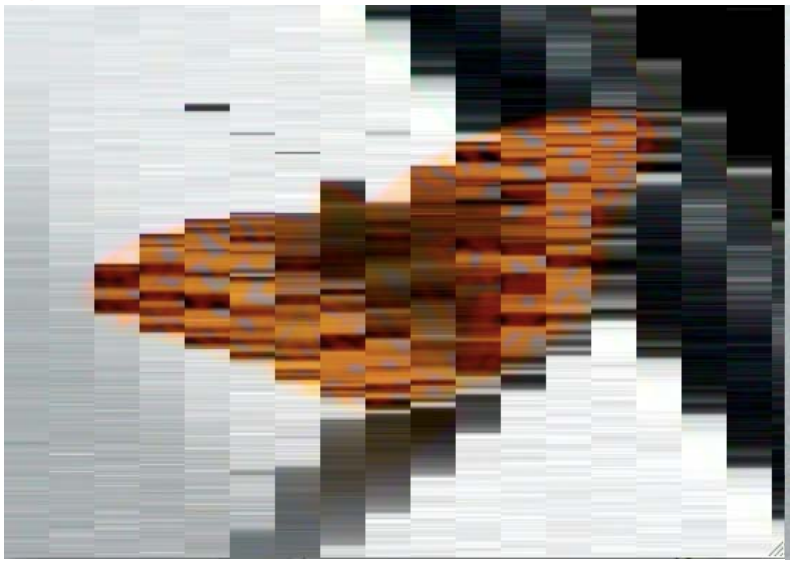


Luminancja (Y) co 16 wartość, U i V bez zmian

Chrominancja (U i V) co 16 wartość, Y bez zmian

ORYGINAŁ

Kolor



Luminancja (Y) co 64 wartość, U i V bez zmian

Chrominancja (U i V) co 64 wartość, Y bez zmian

ORYGINAŁ

Kolor

W internecie znajdziemy wiele przykładowych kodów umożliwiających przeliczenie wartości pomiędzy systemami kolorów (np. RGB->YUV->RGB).

Podsumowując należy wskazać najbardziej istotne klasy w Javie związane z kolorem:

- java.awt.color.ColorSpace – obsługa systemów kolorów,
- java.awt.Color – reprezentacja koloru,
- java.awt.image.ColorModel – model koloru dla potrzeb definiowania obrazów cyfrowych

Kolor

Klasa ColorSpace umożliwia operacje na systemie kolorów. Klasa Color obsługuje standardowy model RGB – opisany wcześniej. Natomiast nieco uwagi wymaga klasa ColorModel. Warto zauważyć, że została ona zdefiniowana w pakiecie image. Klasa ColorModel opisuje bowiem konkretną metodę odwzorowania wartości piksela obrazu na dany kolor.
Najbardziej znane formy zapisu komponentów to tablice komponentów (czyli 4 macierze, po jednej dla przezroczystości A i RGB) lub jedna wartość 32 bitowa, gdzie każde 8 bitów tej liczby wykorzystywane jest do przechowywania informacji o danym komponencie (najstarsze 8 bitów to A, potem kolejno RGB).

Wykorzystywane podklasy klasy abstrakcyjnej ColorModel to: ComponentColorModel, IndexColorModel oraz PackedColorModel.

Język JAVA – kolor

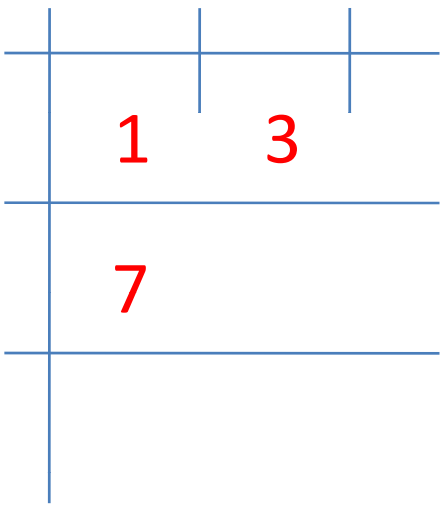
Kolor

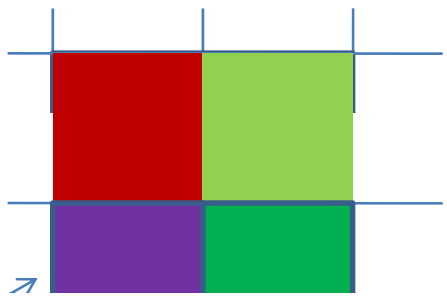
ComponentColorModel - każda próbka (komponent) jest przechowywana oddzielnie. Dla każdego piksela tworzony jest zbiór odseparowanych próbek (danego systemu kolorów, np. R G B). Kolejność występowania próbek jest definiowana przez przestrzeń kolorów, przykładowo dla przestrzeni RGB - TYPE_RGB, index 0 oznacza próbkę dla komponentu RED, itd.

Typ danych dla przechowywania pojedynczych próbek może być 8, 16 lub 32-bitowy:
DataBuffer.TYPE_BYTE,
DataBuffer.TYPE_USHORT,
DataBuffer.TYPE_INT.

Kolor

IndexColorModel - odwołuje się do koloru nie poprzez zbiór próbek lecz poprzez wskaźnik do tablicy kolorów. W indeksowanej tablicy kolorów mamy zdefiniowany zestaw kolorów (np. RGB(100, 230,340)). Indeks do tablicy oznacza zatem kolor. W macierzy danych obrazu wartości traktowane są jako indeksy do tablicy kolorów. Typowe zastosowanie – skala szarości w medycynie.





Kolor

PackedColorModel - jest abstrakcyjną klasą w ramach której kolor jest oznaczany dla danej przestrzeni kolorów poprzez definicję wszystkich komponentów oddzielnie, spakowanych w jednej liczbie 8, 16 lub 32 bitowej. Podklasą tej abstrakcyjnej klasy jest DirectColorModel. Przy wywołaniu konstruktora podaje się liczbę bitów na piksel oraz wartości masek dla każdego komponentu celem wskazania wartości tych komponentów.

Jak widać sposób modelowania koloru istotnie wpływa na metodę reprezentacji obrazów rastrowych. Obraz cyfrowy w modelu rastrowym będzie się składał z macierzy wartości próbek (dane obrazu) oraz systemu (np. RGB) i modelu koloru (ComponentColorModel).

Plan wykładu:

1. Wprowadzenie do grafiki w Javie
2. Budowa GUI: komponenty, kontenery i układanie komponentów
3. Budowa GUI: obsługa zdarzeń
4. Grafika wektorowa – porysujmy sobie
5. Reprezentacja koloru
6. Grafika rastrowa - obrazy
7. Obsługa czcionek