Zespół4_etap1

POLAK MACIEJ, PRANICA JAKUB, SZEWCZYK BARBARA, TROJAN ROBERT

Treść zadania :

Należy utworzyć specyfikację **rastrowego** pliku graficznego rejestrującego obraz **kolorowy** (z wykorzystaniem 32 narzuconych oraz 32 dedykowanych barw) i w **32 stopniowej skali szarości** we wszystkich przypadkach opierającego się na dowolnej kompresji (do wyboru - w projekcie wymagana jest implementacja tylko jednego z algorytmów: ByteRun, RLE, LZW lub LZ77) zapisującego dane z wykorzystaniem jednego z filtrów (**predykatorów**) podobnie jak ma to miejsce w formacie PNG. Do zapisu nieskompresowanych punktów obrazu wykorzystywane jest **5 bitów** na piksel. Należy napisać aplikacje, które zgodnie ze stworzoną specyfikacją dokonają konwersji z pliku BMP do nowego rodzaju pliku graficznego (z możliwością wyboru trybu: kolor lub skala szarości) oraz z nowego rodzaju pliku do formatu BMP.

Specyfikacja projektu w formie tabeli:

	Rozmiar	Opis
Nagłówek		
signature	2 bajty	‚ST’
Informacje Nagłówkowe		
width	4 bajty	Szerokość bitmapy
height	4 bajty	Wysokość bitmapy
colorMode	4 bajty	0 jeśli szarość, 1 jeśli narzucone, 2 jeśli dedykowane
Kolory		
Dedykowane kolory		Paletę dedykowaną zapisywana jest w postaci mapy, dodawana tylko, gdy „ colorMode == 2”
Predyktor		
Używany filtr		Filtr różnicy linii
Kompresja		
Używana kompresja		RLE

Struktura w programie:

```
typedef struct PSKRINFOHEADER {
   char signature[2] = {'S','T'};
   int width;
   int height;
   int colorMode; //0=skala szarosci, 1=kolory narzucone, 2=paleta dedykowana
} PSKRINFOHEADER;

struct RGB {
   Uint8 r;
   Uint8 g;
   Uint8 b;
};

std::map<Uint8,RGB> paletaDedykowana;
```

Obliczanie palety:

Zapis do pliku:

```
SDL_Color color = getPixel(i, j);

/* zamiana piksela z 24 na 5bit*/
RGB c;
Uint8 pixel;

if(colorMode==0){ //skali szarości:
                        pixel = 0.299*color.r+0.587*color.g+0.114*color.b;
                        pixel = pixel >> 3; //5bitowa skala szarości
}else{ //kolor:
                        c.r = color.r >> 6; //zamiana na wartość 2-bitową (000000RR)
                        c.g = color.g >> 6; //zamiana na wartość 2-bitową (000000GG)
                        c.b = color.b >> 7; //zamiana na wartość 1-bitową (0000000G)
                        pixel = c.r | (c.g<<2) | (c.b<<4); //zapis wszystkich składowych na 1 bajcie
aby łatwo zapisać do pliku (000BGGRR)
}

plik->write(  reinterpret_cast < char *>( &pixel ),  sizeof(pixel)); //zapis do pliku
```

Wyświetlanie:

```
//Zmienna pixel to jeden bajt odczytywany z pliku. Poszczególne składowe zapisane są na
następujących bitach: 000BGGRR
Uint8 pixel;
plikde.read( reinterpret_cast < char *>( &pixel ), sizeof(pixel) );
if(header.colorMode == 2){ //tryb kolorow dedykowanych:
        RGB kolorDedykowany = paletaDedykowana[pixel];
        setPixel(i, j, kolorDedykowany.r, kolorDedykowany.g, kolorDedykowany.b);          }
else if(header.colorMode == 1){ //tryb kolorow narzuconych:
         RGB color;

                        //wydzielamy poszczególne składowe:
                        color.r = pixel & 0x03;
                        color.g = (pixel & 0x0C) >> 2;
                        color.b = (pixel & 0x10) >> 4;

                        color.r *= 85; //równoważne z działaniem: R/3*255
                        color.g *= 85;
                        color.b *= 255;
                        setPixel(i, j, color.r, color.g, color.b);
} else{ //tryb stali szarosci:
                        pixel = pixel/31.0*255.0;
        setPixel(i, j, pixel,pixel,pixel);
}
```