2. Spaţiul culorilor. Conversii color ⇒ grayscale şi grayscale ⇒ alb-negru

2.1. Introducere

Scopul acestui al doilea laborator este învăţarea tehnicilor de bază în lucrul cu culorile imaginilor digitale în format bitmap.

2.2. Spaţiul RGB

Culoarea fiecărui pixel (atât pentru echipamentele de achiziţie – camere) cât şi pentru afişare (TV, CRT, LCD) se obţine prin combinaţia a trei culori primare: roşu, verde şi albastru. (**R**ed, **G**reen şi **B**lue.) (spaţiu de culoare aditiv – fig. 2.1 şi 2.2).

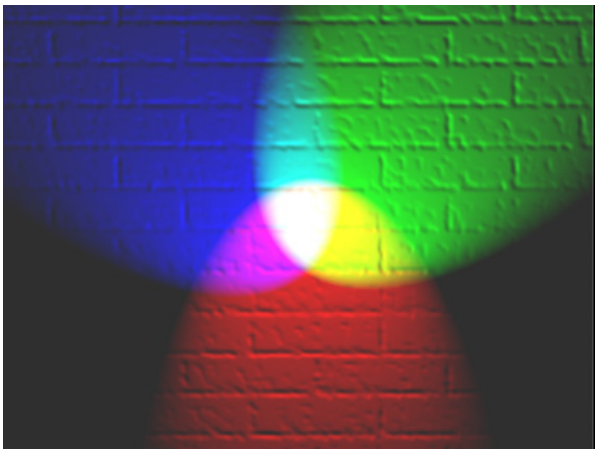


Fig. 2.1. Reprezentarea combinării aditive a culorilor. Acolo unde culorile primare se suprapun se observă apariţia culorilor secundare. Acolo unde toate trei culorile se suprapun se observă apariţia culorii albe[1].

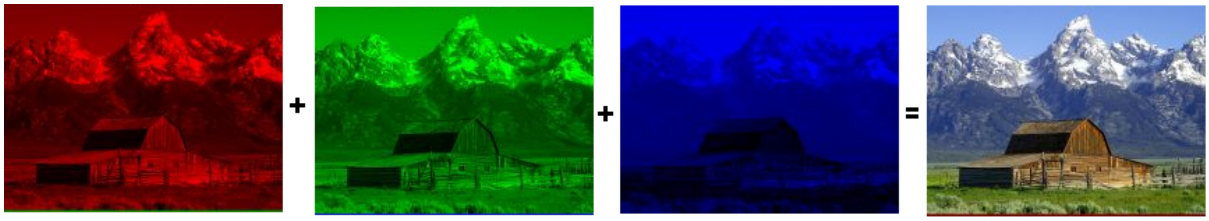


Fig. 2.2. Imaginea color se obţine prin combinarea la nivel de pixel a celor trei culori primare (vezi cele trei canale).

Astfel, fiecare pixel din imagine va fi caracterizat prin câte o valoare pentru fiecare din cele trei componente de culoare primare. Culoarea sa reprezintă un punct în spaţiul 3D al modelului de culoare RGB (fig. 2.3). În acest cub al culorilor, originea axelor R, G şi B corespunde *culorii negre* (0, 0, 0). Vârful opus al cubului corespunde *culorii albe* (255, 255, 255). Diagonala cubului, între negru şi alb corespunde tonurilor de gri (grayscale) (R=G=B). Trei dintre vârfuri corespund culorilor primare roşu, verde şi albastru. Celelalte 3 vârfuri corespund culorilor complementare: turcoaz, mov şi galben (**C**yan, **M**agenta and **Y**ellow). Dacă translatăm originea sistemului de coordonate în punctul „alb” şi redenumim cele 3 axe de coordonate ale sistemului în C, M, Y obţinem spaţiul de culoare complementar CMY, (folosit la dispozitive de imprimare color).

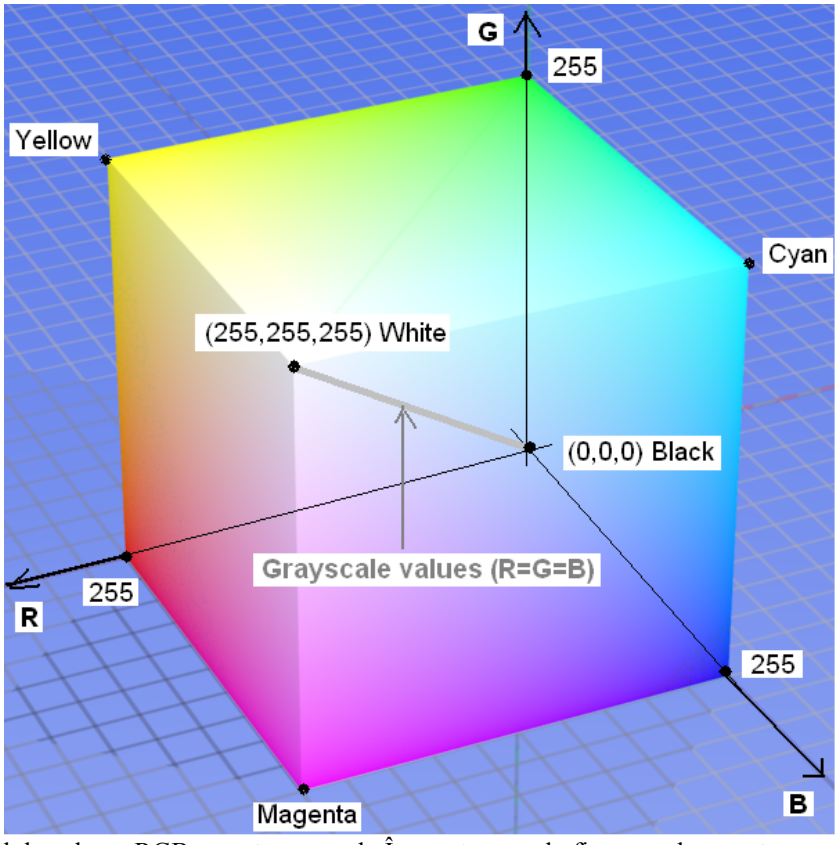


Fig. 2.3. Modelul de culoare RGB mapat pe un cub. În acest exemplu fiecare culoare este reprezentată pe câte 8 biţi (256 de nivele) (imagini bitmap RGB24). Numărul total de culori este 2^8x2^8x2^8 = 2^{24} = 16.777.216.

Pentru imagini RGB24 (24 biţi/pixel) spaţiul de culoare poate fi reprezentat complet. Într-o imagine indexată (cu paletă) poate fi reprezentat doar un anumit subspaţiu al spaţiului de culoare din figura 2.3. În acest context, numărul de biţi/pixel (numărul de biţi folosiţi pentru codificarea unei culori) se numeşte „adâncime de culoare” (color depth). (Tabelul 2.1):

Tabel 2.1. Adâncimea şi tipul imaginii

Adâncimea de culoare	**Nr. Culori**	**Mod de culoare**	**Palette (LUT)**
1 bit	2	Indexed Color	Yes
4 biţi	16	Indexed Color	Yes
8 biţi	256	Indexed Color	Yes
16 biţi	65536	True Color	No
24 biţi	16.777.216	True Color	No
32 biţi	16.777.216	True Color	No

Există şi alte modele de culoare[2] care nu vor fi discutate aici.

2.3 Conversia unei imagini color într-o imagine grayscale

Pentru a converti o imagine color intr-o imagine grayscale, cele trei componente ale culorii fiecărui pixel trebuie egalizate. O metodă des folosită este medierea celor 3 componente:

$$R_{Dst} = G_{Dst} = B_{Dst} = \frac{R_{Src} + G_{Src} + B_{Src}}{3} \qquad (2.1)$$

2.3.1. Cazul imaginilor RGB24 (24 biţi/pixel)

În acest caz formula 2.1 se poate aplica accesând, fiecare dintre cele trei componente de culoare ale fiecărui pixel din imaginea sursă şi destinaţie, aşa cum a fost explicat în laboratorul 1.

2.3.2. Cazul imaginilor indexate (cu paletă)

În acest caz, intrările din paleta imaginii destinaţie trebuie parcurse (vezi exemplul din laboratorul 1) iar componentele culorilor din fiecare intrare trebuie convertite folosind (2.1). O situaţie des întâlnită după această operaţie simplă este faptul că intrările paletei nu sunt ordonate crescător, după nivelul de gri. (fig. 2.4).

Index vechi	R	G	B	X
0	100	100	100	-
1	20	20	20	-
2	32	32	32	-
.				
.				
.				
255	78	78	78	-

Fig. 2.4. Paleta nesortată după convertirea din color la grayscale în cazul imaginilor indexate.

Alte procesări ulterioare ale imaginii grayscale necesită o paletă ordonată. Din acest motiv, după operaţia de conversie trebuie efectuată o operaţie de ordonare a paletei.

Metodă simplă de ordonare a paletei:

1. Se creează un vector de dimensiune 256:

```
ex: BYTE g[256];
```

2. Se parcurge paleta şi se iniţializează elementele lui *g* cu una dintre componentele de culoare ale intrării *k* din paleta grayscale nesortată (fig. 2.4) urmată de „sortarea" paletei prin atribuirea valorii *k* la cele 3 componente de culoare ale intrării *k* (în această ordine):

```
for (k = 0 … iColors) {
     // initializare vector g
     g[k] = paleta[k].rgbRed;
     // „sortare" paleta
     paleta[k].rgbRed = paleta[k].rgbGreen = paleta[k].rgbBlue = k;
}
```

Index nou	R	G	B	X	g
0	0	0	0	-	5
1	1	1	1	-	23
2	2	2	2	-	14
.					
.					
.					
255	255	255	255	-	243

Fig. 2.5. Paletă sortată după pasul 2.

3. Un ultim pas constă în parcurgerea pixelilor din imagine şi înlocuirea valorilor (indecşilor) vechi *k* cu valoarea *g*[*k*] (folosind corespondenţele din vectorul *g* obţinute la pasul anterior). (fig. 2.5):

```
k = lpDst[i*w+j];
lpDst[i*w+j] = g[k];
```

2.4. Accesarea conţinutului paletei şi a altor informaţii relevante din header-ul bitmap

Următorul exemplu arată cum poate fi accesat header-ul bitmap:

```
//Obţine pointer-ul la începutul header-ului bitmap
//tipul pointer-ului este BITMAPINFO*
LPBITMAPINFO pBitmapInfoSrc = (LPBITMAPINFO)lpS;
```

sau

```
BITMAPINFO *pBitmapInfoSrc = (BITMAPINFO*) lpS;

// obţine dimensiunea bitmap-ului
pBitmapInfoSrc->bmiHeader.biSize;
//obţine numărul de biţi/pixel
pBitmapInfoSrc->bmiHeader.biBitCount); //numărul de biţi/pixel (1, 4, 8, 16,
//24, 32
.........
```

Unde structurile BITMAPINFO şi BITMAPINFOHEADER sunt definite ca:

```
typedef struct tagBITMAPINFO {
  BITMAPINFOHEADER bmiHeader;
  RGBQUAD          bmiColors[1];
} BITMAPINFO, *LPBITMAPINFO;

typedef struct tagBITMAPINFOHEADER{
  DWORD  biSize;
  LONG   biWidth;
  LONG   biHeight;
  WORD   biPlanes;
  WORD   biBitCount;
  DWORD  biCompression;
  DWORD  biSizeImage;
  LONG   biXPelsPerMeter;
  LONG   biYPelsPerMeter;
  DWORD  biClrUsed;
  DWORD  biClrImportant;
} BITMAPINFOHEADER, *LPBITMAPINFOHEADER;
```

Modul în care se pot accesa intrările din paletă a fost prezentat în laboratorul 1.

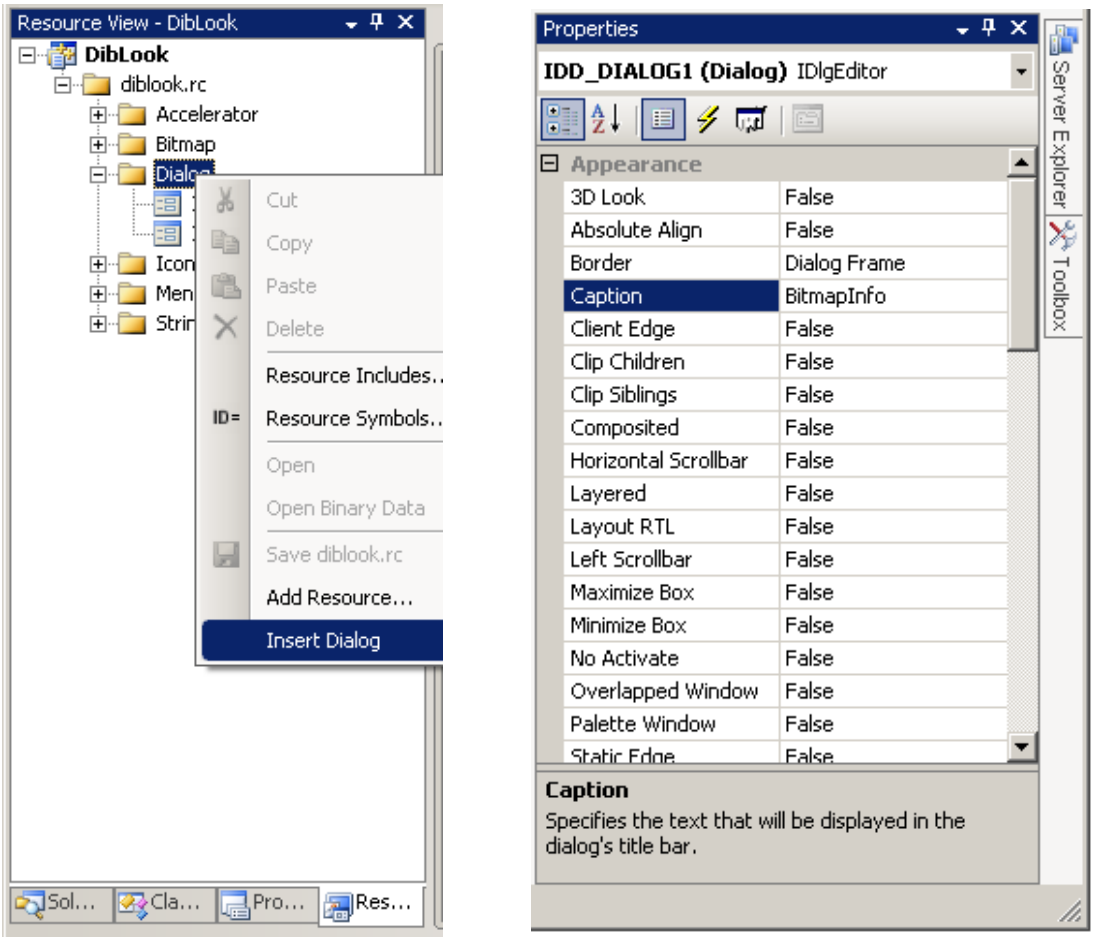
2.5. Scurt ghid despre afişarea informaţiilor într-o căsuţă de dialog

2.5.1. Crearea unei noi resurse de tip *Dialog Box*

1. În fereastra *Workspace* comutaţi pe tabulatorul *Resource View*, selectaţi intrarea *Dialog* faceţi click dreapta şi click stânga pe *Insert Dialog* (fig. 2.6.a).

2. Faceţi click (selectaţi) resursa de tip dialog nou creată. În dreapta se va activa fereastra *Properties* (fig. 2.6.b) asociata dialogului respectiv. Puteţi schimba numele (se recomandă să o faceţi), stilul şi ID-ul (nu se recomandă) şi aşa mai departe …

3. Faceţi click dreapta pe noua resursă dialog creată (fig. 2.7) si selectaţi opţiunea pentru a crea o nouă clasă pentru noua resursă dialog (‘*Add class*’). Se va deschide fereastra „*MFC Class Wizard*” (fig. 2.8)



a. b.

Fig. 2.6.

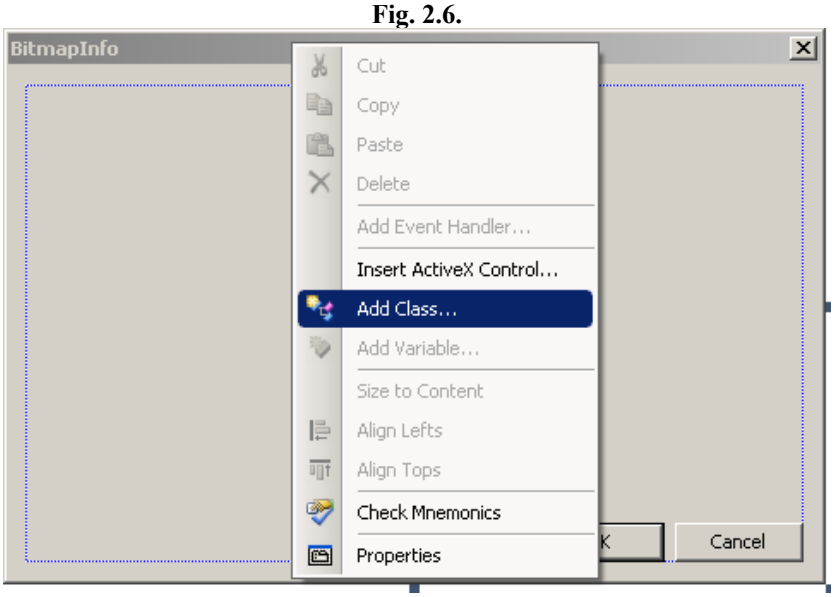


Fig. 2.7.

4. Daţi un nume relevant clasei asociate dialogului (de exemplu CBitmapInfoDlg.) Wizard-ul creează automat fişierele necesare *.h and *.cpp noii clase (numele fişierului este similar

numelui clasei, şi nu este necesar să îl modificaţi). Noua clasă poate fi accesată cu uşurinţă din tabulatorul *ClassView* al ferestrei *Workspace* (în care este adăugată în mod automat – fig. 2.10.d).

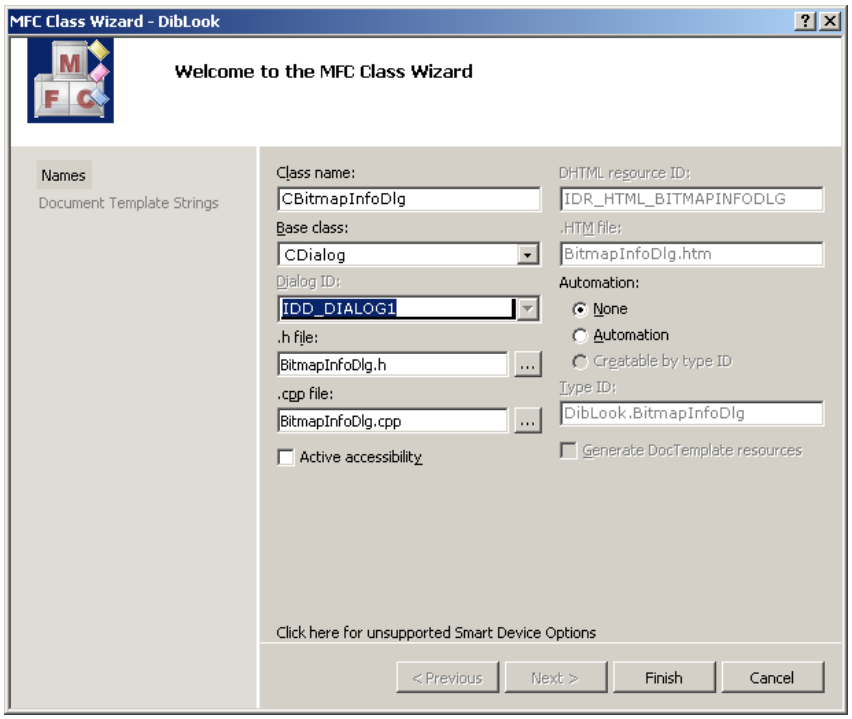


Fig. 2.8.

5. Includeţi fişierul header al noului dialog în secţiunea *#include* a fişierului *dibview.cpp*:

```
#include "BitmapInfoDlg.h"
```

6. Creaţi o instanţă (obiect) al clasei şi afişaţi dialogul în funcţia de procesare dorită. Funcţia de mai jos afişează dialog-ul doar în mod modal (codul ulterior apelul metodei *DoModal()* va fi executat doar după ce dialogul va fi închis). Dialog-urile se pot afişa şi non-modal (temă pentru cine doreşte)

```
void CDibView::OnProcessingAfisarebmpheader()
{
	// Puteti folosi acest apel de macro in cazul in care
	// nu aveti nevoie sa afisati o imagine rezultat
	BEGIN_SOURCE_PROCESSING;

	//creare a unei instaţe (obiect) a clasei dialog
	CBitmapInfoDlg dlgBmpHeader;

	// Adaugaţi aici cod pentru citirea header-ului bitmap şi pentru scrierea
	// lui în dialog

	//afişează dialogul în mod modal
	dlgBmpHeader.DoModal();

	// Puteti folosi acest apel de macro in cazul in care
	// nu aveti nevoie sa afisati o imagine rezultat
	END_SOURCE_PROCESSING;
}
```

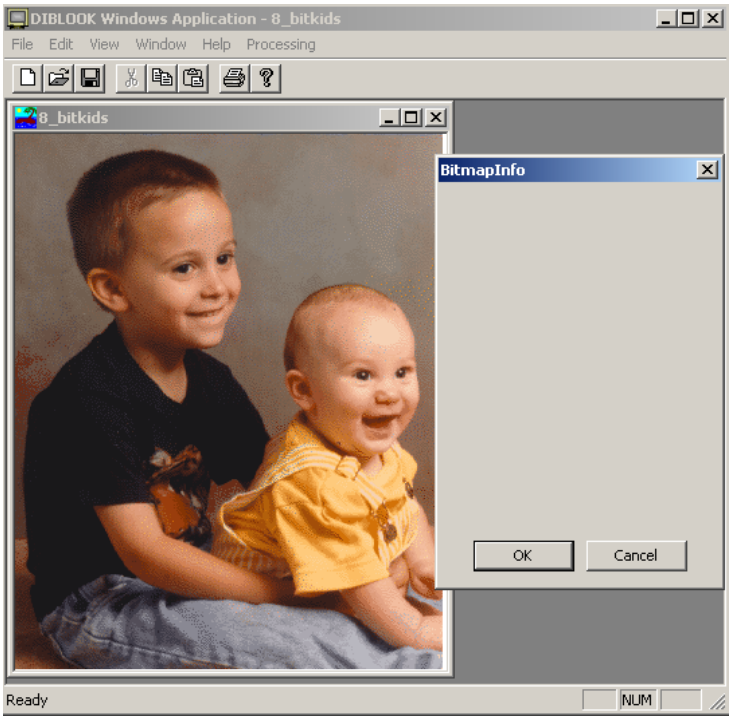


Fig. 2.9.

2.5.2. Proiectarea căsuţei de dialog

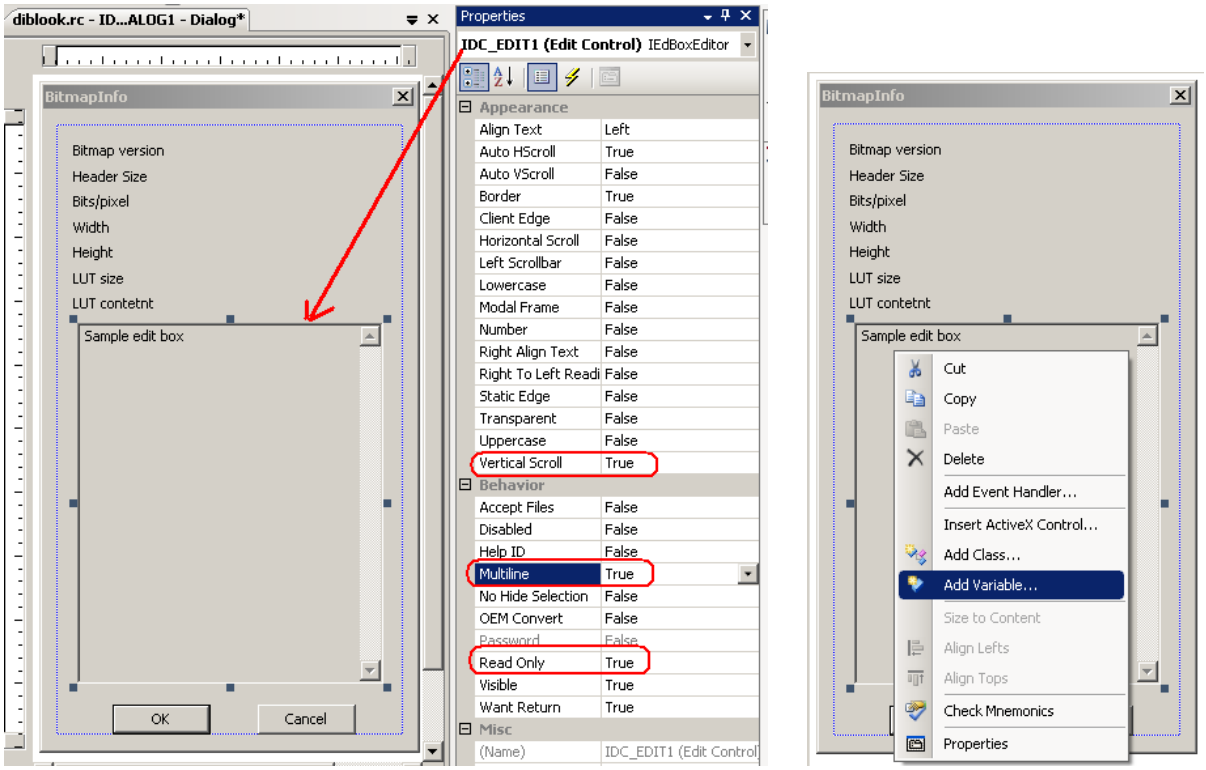
Pentru a afişa sau a obţine date de la utilizator folosind căsuţa de dialog, trebuie adăugate controale la resursa dialog creată anterior. Selectaţi fereastra *Toolbox* (din meniul *View->Toolbox*), selectaţi un control şi apoi faceţi click pe dialog în zona în care doriţi să îl plasaţi. Cel mai folosite tipuri de controale sunt *Static Text* (pentru afişare) şi *Edit Control* (pentru afişare/introducere). În cazul de faţă avem nevoie doar de afişare.

Câmpurile individuale (cum ar fi înălţimea şi lăţimea bitmap-ului) pot fi afişate cu uşurinţă în controale de tip S*tatic Text*. Pentru a afişa ceva într-un control de tip static text, acesta trebuie să aibă un ID individual dat explicit (id-ul implicit este ID_STATIC şi este comun pentru toate controalele static text, deci trebuie specificate explicit ID-uri unice (ID_STATIC1 …).

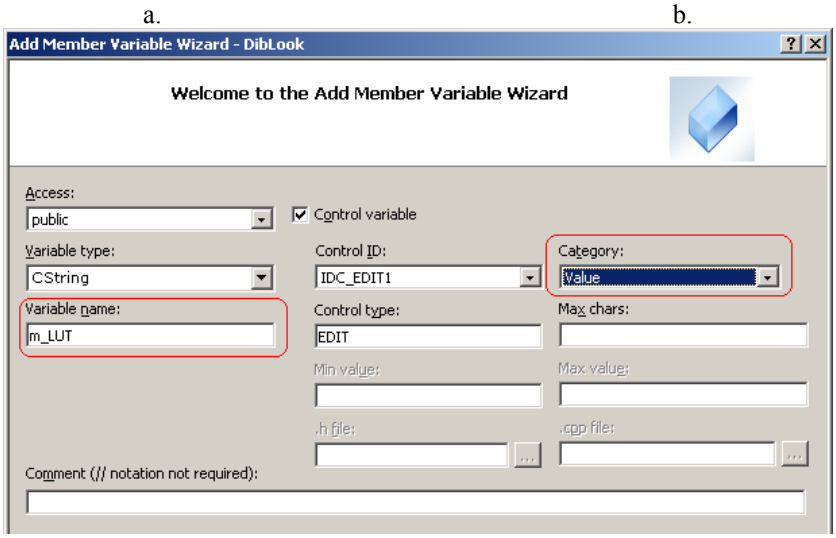
Tablele (de exemplu conţinutul paletei) pot fi afişate folosind un *Edit Control*. Controalele *Edit Control* au implicit alocat câte un ID individual (nu este necesar să îl modificaţi). Pentru controalele de tip *Edit Control* stilul poate fi editat în funcţie de modul de afişare dorit (fig. 2.10.a).

Odată ce controalele au fost adăugate la dialog, fiecăruia dintre ele trebuie să îi fie asociată o variabilă. Aceasta se poate face deschizând dialogul *Add Member Variable Wizard* (se face click dreapta pe resursa dialog şi se selectează opţiunea *Add Variable* (fig. 2.10.b).

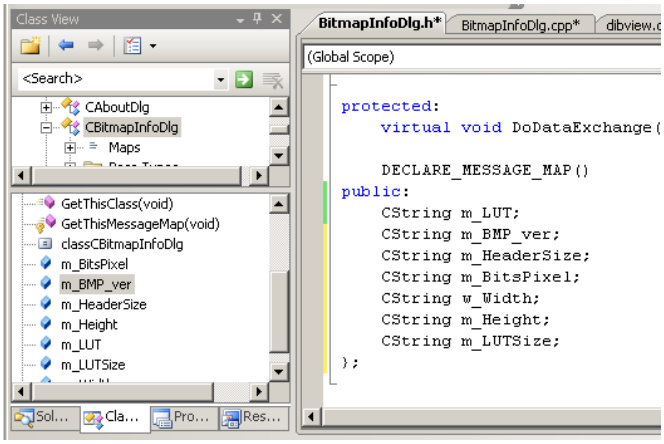
În dialogul *Add Member Variable Wizard* (fig. 2.10.c) pentru fiecare control, (identificat prin ID) se poate adăuga o variabilă membru (se specifică numele, tipul (selectaţi *CString*), categoria (selectaţi *Value*), etc.). Variabilele membru asociate controalelor folosind *Add Member Variable Wizard* sunt adăugate automat clasei dialog (fig. 2.10.d).


diblook.rc - ID...ALOG1 - Dialog*
BitmapInfo
Bitmap version
Header Size
Bits/pixel
Width
Height
LUT size
LUT contetnt
Sample edit box
OK
Cancel
Properties
IDC_EDIT1 (Edit Control) IEdBoxEditor
Appearance
Align Text Left
Auto HScroll True
Auto VScroll False
Border True
Client Edge False
Horizontal Scroll False
Left Scrollbar False
Lowercase False
Modal Frame False
Number False
Right Align Text False
Right To Left Readi False
Static Edge False
Transparent False
Uppercase False
Vertical Scroll True
Behavior
Accept Files False
Disabled False
Help ID False
Multiline True
No Hide Selection False
OEM Convert False
Password False
Read Only True
Visible True
Want Return True
Misc
(Name) IDC_EDIT1 (Edit Control)
BitmapInfo
Bitmap version
Header Size
Bits/pixel
Width
Height
LUT size
LUT contetnt
Sample edit box
Cut
Copy
Paste
Delete
Add Event Handler...
Insert ActiveX Control...
Add Class...
Add Variable...
Size to Content
Align Lefts
Align Tops
Check Mnemonics
Properties

a. b.


Add Member Variable Wizard - DibLook
Welcome to the Add Member Variable Wizard
Access:
public
Control variable
Variable type:
CString
Control ID:
IDC_EDIT1
Category:
Value
Variable name:
m_LUT
Control type:
EDIT
Max chars:
Min value:
Max value:
.h file:
.cpp file:
Comment (// notation not required):

c.


Class View
<Search>
CAboutDlg
CBitmapInfoDlg
Maps
GetThisClass(void)
GetThisMessageMap(void)
classCBitmapInfoDlg
m_BitsPixel
m_BMP_ver
m_HeaderSize
m_Height
m_LUT
m_LUTSize
BitmapInfoDlg.h*
BitmapInfoDlg.cpp*
(Global Scope)
protected:
virtual void DoDataExchange(
DECLARE_MESSAGE_MAP()
public:
CString m_LUT;
CString m_BMP_ver;
CString m_HeaderSize;
CString m_BitsPixel;
CString w_Width;
CString m_Height;
CString m_LUTSize;
};

d

Fig. 2.10.

2.5.3. Afişarea în căsuţa de dialog

Pentru a afişa datele dorite în căsuţa de dialog, datele trebuie scrise în variabilele asociate controalelor de pe dialog. Această scriere trebuie efectuată în funcţia de procesare, (anterior apelului metodei DoModal() care afişează dialogul):

```
void CDibView::OnProcessingAfisarebmpheader()
{
      BEGIN_SOURCE_PROCESSING;

      //crează o instanţă a clase dialog
      CBitmapInfoDlg dlgBmpHeader;
      LPBITMAPINFO pBitmapInfoSrc = (LPBITMAPINFO)lpS;
      dlgBmpHeader.m_Width.Format(_TEXT("Image width [pixels]: %d"),
                                  pBitmapInfoSrc->bmiHeader.biWidth);
      // şi celelalte informaţii ......

      // Stocarea intrărilor din paletă într-o variabilă CString m_LUT
      // (asociată edit control-ului pentru afişarea paletei)
      CString buffer;
      for (int i=0;i<iColors;i++)
      {
            buffer.Format(_TEXT("%3d.\t%3d\t%3d\t%3d\r\n"),i,
                                  bmiColorsSrc[i].rgbRed,
                                  bmiColorsSrc[i].rgbGreen,
                                  bmiColorsSrc[i].rgbBlue);
            dlgBmpHeader.m_LUT+=buffer;
      }
      //afişarea dialogului în mod modal
      dlgBmpHeader.DoModal();

      END_SOURCE_PROCESSING;
}
```

2.6. Conversia imaginilor grayscale în imagini binare (alb-negru)

O imagine binară (alb-negru) este o imagine care conţine doar două culori: alb şi negru. O imagine binară se obţine dintr-o imagine grayscale printr-o operaţie simplă numită binarizare cu prag (thresholding). Binarizarea cu prag este cea mai simplă tehnică de segmentare a imaginilor, care permite separarea obiectelor de background. (fig. 2.11).

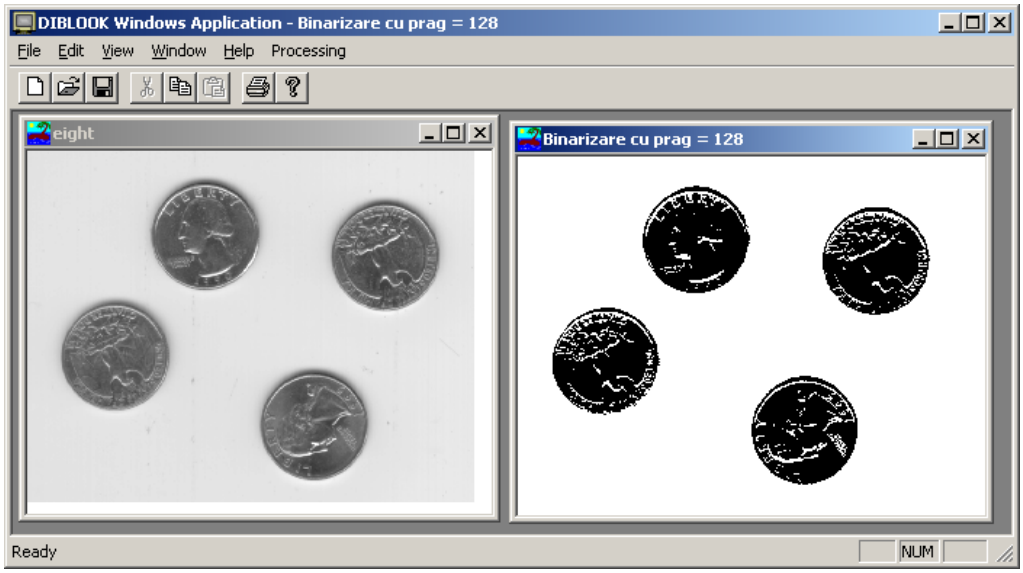


Fig. 2.11.

În acest laborator va fi discutată binarizarea cu prag fix (ales arbitrar) pentru imagini indexate (8 biţi/pixel) de tip grayscale. Binarizarea poate fi aplicată prin parcurgerea valorilor pixelilor din imaginea sursă şi înlocuirea lor în imaginea destinaţie cu valoarea dată de:

$$lpDst[i*w+j] = \begin{cases} 0 \quad (black) & , \quad if \quad lpSrc[i*w+j] < threshold \\ 255 \quad (white) & , \quad if \quad lpSrc[i*w+j] \geq threshold \end{cases} \tag{2.2}$$

Valoarea pragului poate fi stabilită inline (în cadrul codului) (nerecomandat) sau printr-o căsuţă de dialog (recomandat). Crearea acestei resurse de dialog şi adăugarea la ea a unui control de tip *Edit Control* este similară cu ceea ce s-a discutat în secţiunea 2.5. *Edit Control*-ul trebuie să permită editarea conţinutului său (adică să fie non-read-only, implicit), asa cum e arătat în figura 2.12. Tipul variabilei folosite pentru a obţine sau a modifica valoarea introdusă în căsuţa de dialog poate fi unul numeric (BYTE) (fig. 2.12).

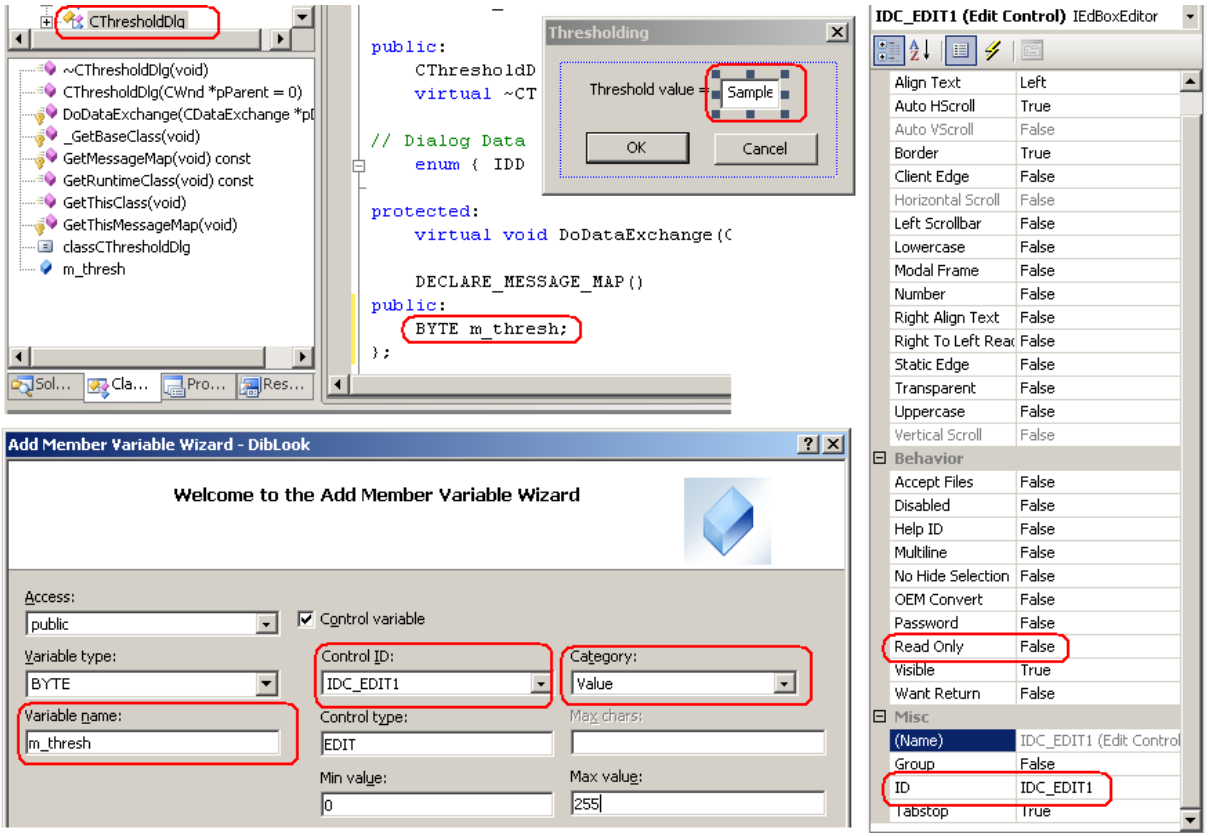


Fig. 2.12.

Cod demonstrativ pentru obţinerea valorii threshold-ului din dialog:

```
void CDibView::OnProcessingBinarizarecupragarbitrar()
{
      BYTE threshold;

      //crează o instanţă (obiect) al clasei dialog
      CThresholdDlg dlgThresh;

      if (dlgThresh.DoModal() == IDOK) {
            threshold=dlgThresh.m_thresh;

            INCEPUT_PRELUCRARI();
            //  Parcurgere pixeli şi aplicare prag.
            // ...
```

```
                CString buf;
                buf.Format(_TEXT("Binarizare cu prag = %d"), threshold);
                SFARSIT_PRELUCRARI(buf);
        }

}
```

2.7. Activităţi practice

1. Adăugaţi la framework-ul DIBLook o funcţie pentru afişarea (într-o căsuţă de dialog) a informaţiilor din header-ul bitmap şi a conţinutului paletei.
2. Adăugaţi la framework-ul DIBLook o funcţie de conversie de la color la grayscale pentru imagini RGB24 (24 biţi/pixel), folosind (2.1).
3. Adăugaţi la framework-ul DIBLook o funcţie de procesare pentru conversia de la color la grayscale pentru imagini indexate (8 biţi/pixel), folosind (2.1).
4. Adăugaţi la framework-ul DIBLook o funcţie de procesare care sortează paleta unei imagini indexate, ca în secţiunea 2.3.2.
5. Comparaţi conţinutul unei palete sortate cu cel al unei palete nesortate (folosind imaginea grayscale obţinută din imaginea color *Kids.bmp*).
6. Integraţi funcţiile de la punctele 3 şi 4 într-una singură.
7. Adăugaţi la framework-ul DIBLook o funcţie de procesare pentru conversia de la *grayscale* la *alb-negru* pentru imagini indexate (8 biţi/pixel), folosind (2.2). Citiţi valoarea pragului folosind o căsuţă de dialog. Testaţi operaţia de binarizare folosind diverse imagini si diverse praguri.
8. **Salvaţi-vă ceea ce aţi lucrat. Utilizaţi aceeaşi aplicaţie în laboratoarele viitoare. La sfârşitul laboratorului de procesare a imaginilor va trebui să prezentaţi propria aplicaţie cu algoritmii implementaţi!!!**

Referinţe

[1] http://en.wikipedia.org/wiki/RGB_color_model
[2] http://en.wikipedia.org/wiki/Color_models
[3] http://msdn2.microsoft.com/en-us/library/ms779712(VS.85).aspx