3. Histograma nivelurilor de intensitate

3.1. Introducere

În această lucrare se vor prezenta conceptul de histogramă a nivelurilor de gri şi un algoritm pentru a diviza această histogramă în mai multe zone în scopul reducerii numărului de niveluri de gri (cuantizare în spațiul culorilor).

3.2. Histograma nivelurilor de intensitate

Având o imagine grayscale cu L niveluri de intensitate (pentru o imagine cu 8 biți/pixel $L=255$), histograma nivelelor de intensitate (gri) se defineşte ca o funcție $h(g)$ care are ca valoare, numărul de pixeli din imagine (sau dintr-o regiune) care au intensitatea $g \in [0 \dots L]$, :

$$h(g) = N_g \tag{3.1}$$

N_g - numărul de pixeli din imagine sau din regiunea de interes (ROI) care au intensitatea g.

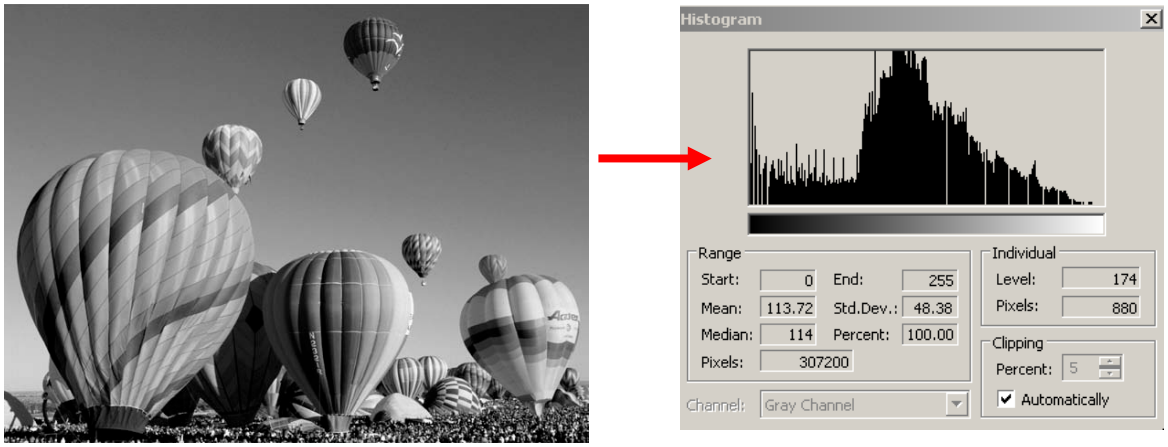


Fig. 3.1 Exemplu: Histogramei unei imagini grayscale

Histograma normalizată cu numărul de pixeli din imagine (din ROI) se numeşte funcția densității de probabilitate (FDP) a nivelelor de intensitate:

$$p(g) = \frac{h(g)}{M} \tag{3.2}$$

unde:

M = *image_height x image_width*

FDP are următoarele proprietăți:

$$\begin{cases} p(g) >= 0 \\ \int_{-\infty}^{\infty} p(g)dg = 1, \quad \sum_{g=0}^{L} \frac{h(g)}{M} = \frac{M}{M} = 1 \end{cases} \tag{3.3}$$

3.3. Aplicație: Determinarea pragurilor multiple

Acest algoritm determină praguri multiple în scopul reducerii numărului de niveluri de intensitate (gri) a unei imagini.

Primul pas constă în determinarea maximelor din histogramă. Apoi fiecare nivel de gri este asignat la cel mai apropiat maxim.

Pentru determinarea maximelor din histogramă se urmăresc pașii:

1. Se normalizează histograma (se transformă într-o FDP)
2. Se alege o fereastră de lățime $2*WH+1$ (o valoare bună pentru WH este 5)
3. Se alege un prag *TH* (o valoare bună este 0.0003)
4. Se „suprapune" fereastra peste histogramă. Pentru fiecare poziție *k* (a mijlocului ferestrei) de la $0+WH$ până la $255-WH-1$
   - Se calculează media *v* a valorilor din histograma normalizată în intervalul $[k-WH, k+WH]$. Obs: valoarea *v* este media a $2*WH+1$ valori
   - Dacă $FDP[k]>v+TH$ și $FDP[k]$ este mai mare sau egal cu toate valorile *FDP* din intervalul $[k-WH, k+WH]$ atunci *k* corespunde la un maxim din histogramă. Se memorează și se continuă din poziția următoare.
5. Se inserează 0 la începutul listei de poziții de maxime și 255 la sfârșit. (aceasta permite culorilor negru respectiv alb să fie reprezentate exact).

Al doilea pas constă în determinarea efectivă a pragurilor. Acestea sunt situate la distanțe egale între maximele determinate anterior. Algoritmul de reducere a nivelurilor de gri este unul simplu: se asignează fiecare pixel la valoarea culorii celui mai apropiat maxim din histogramă.

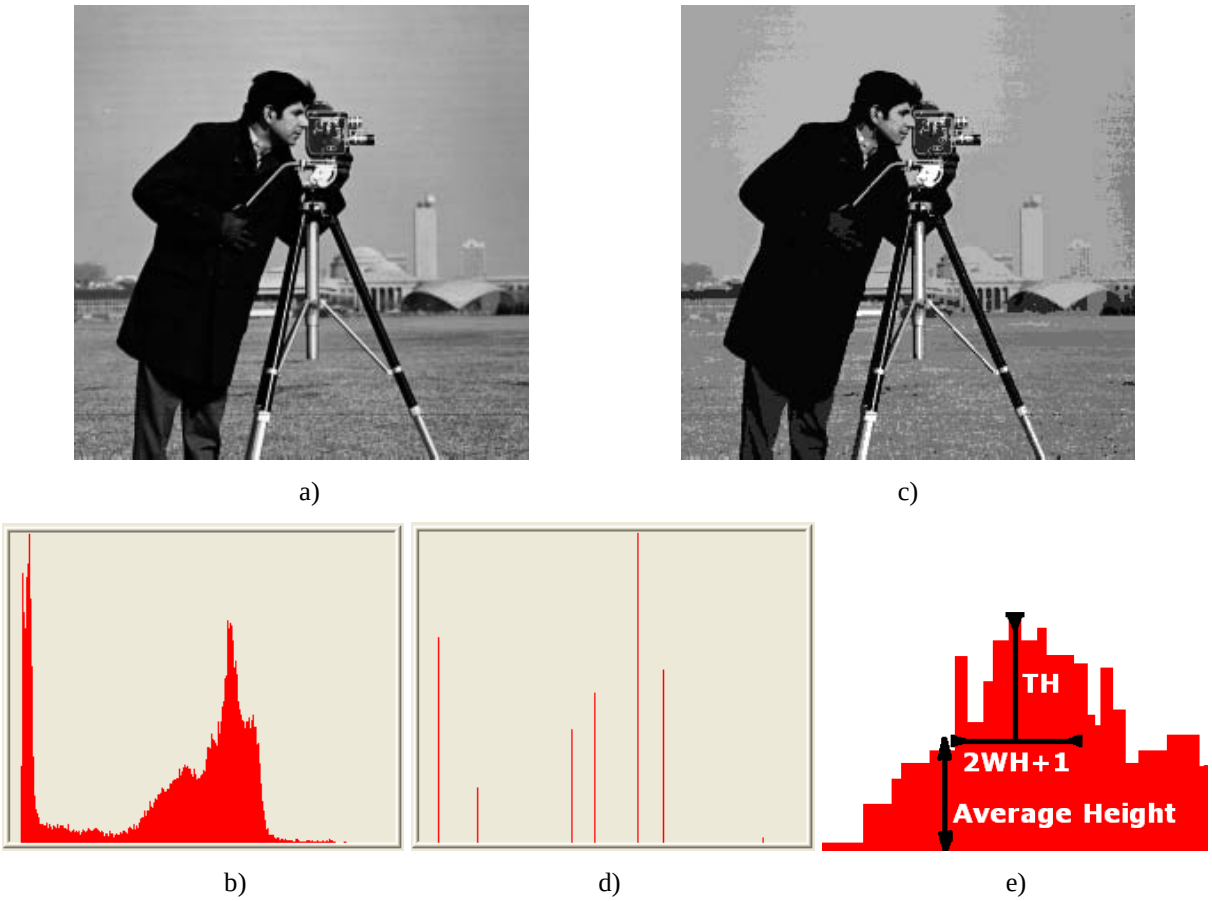


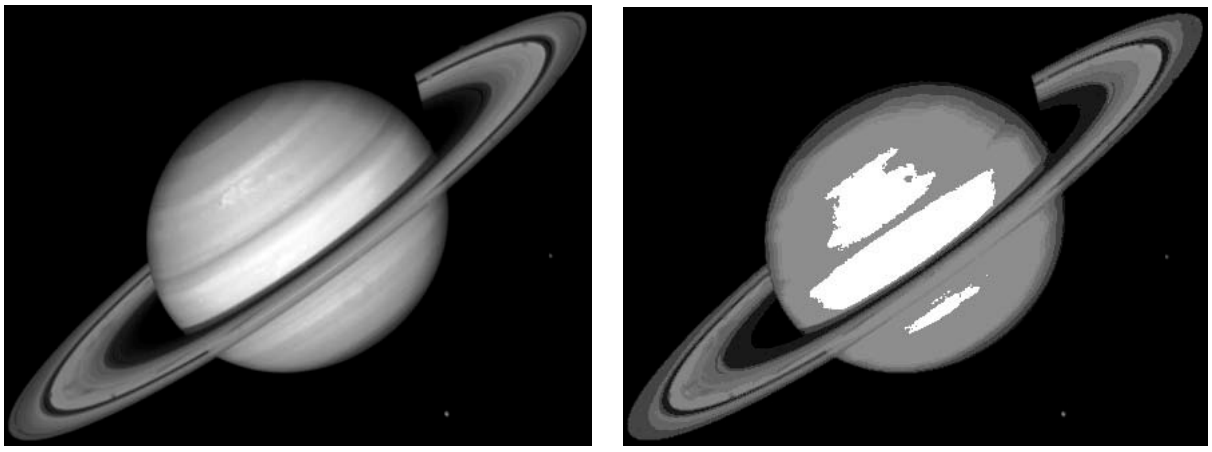
Fig. 3.2 a) Imaginea inițială; b) Histograma imaginii inițiale; c) Imaginea cuantizată obținută (praguri multiple); d) Histograma imaginii cuantizate; e) Algoritmul de calcul al maximelor din histogramă

După cum se observă în Fig. 3.3, rezultatele sunt vizual inacceptabile când numărul de niveluri de gri este scăzut. Pentru a obține rezultate vizuale acceptabile se poate aplica un algoritm de distribuire a erorii de cuantizare pe mai mulți pixeli (*dithering*). Un exemplu de astfel de algoritm este Floyd-Steinberg:

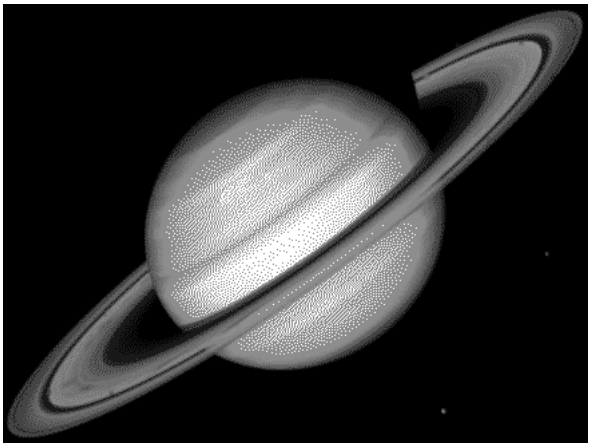
```
pentru fiecare y de jos până sus
   pentru fiecare x de la stânga până la dreapta
      pixel_vechi := sursă(x,y)
      pixel_nou := cel mai apropiat maxim din histogramă
      destinație(x,y) := pixel_nou
      eroare := pixel_vechi - pixel_nou
      sursă(x+1,y) := sursă(x+1,y) + 7*eroare/16
      sursă(x-1,y+1) := sursă(x-1,y+1) + 3*eroare/16
      sursă(x,y+1) := sursă(x,y+1) + 5*eroare/16
      sursă(x+1,y+1) := sursă(x+1,y+1) + eroare/16
```

Acest algoritm calculează eroarea de cuantificare şi o distribuie la pixelii vecini. Fracțiunile de distribuție a erorii sunt prezentate în matricea următoare:

3/16	5/16	1/16
0	0	7/16
0	0	0



a) b)



c)

Fig. 3.3 a) Imaginea inițială; b) Imaginea cuantizată obținută (praguri multiple); c) Împrăștierea erorii pragurilor multiple utilizând algoritmul Floyd-Steinberg

3.4. Detalii de implementare: afişarea histogramei într-o căsuță de dialog

3.4.1. Varianta 1: afişarea histogramei într-un control de tip *Picture* al unei căsuțe de dialog

Histograma nivelurilor de gri se va afişa prin intermediul unei căsuțe de dialog (*Dialog Box*). Se utilizează un control de tip *Picture* din cadrul căsuței de dialog pentru afişarea histogramei.

Controlul de tip *Picture* va avea formă dreptunghiulară şi implicit o anumită lățime şi înălțime. Lățimea acestuia trebuie să fie de minim *L* pixeli unde *L* reprezintă numărul de valori de intensitate corespunzătoare imaginii căreia i se calculează histograma (*L*=256 pentru o imagine grayscale cu 8 biți/pixel). Histograma va fi afişată sub formă de bare verticale de înălțimea corespunzătoare numărului de pixeli pentru fiecare nivel de intensitate. Barele verticale corespunzătoare nivelelor de intensitate *0…L* vor fi afişate de la stânga spre dreapta.

Observație: este posibil ca numărul de pixeli de o anumită intensitate din histogramă să depăşească înălțimea controlului în care se afişează histograma. De aceea se va afişa o histogramă scalată (fiecare valoare din şir va fi divizată cu valoarea maximă din acesta şi apoi înmulțită cu înălțimea controlului de afişare). Scalarea va fi efectuată doar cazul în care maximul din histogramă depăşeşte înălțimea controlului de afişare. Astfel, barele verticale nu vor putea depăşi înălțimea controlului de afişare, păstrându-se raportul dintre valorile din histogramă.

1. Se inserează o nouă căsuță de dialog (vezi laboratorul 2!).
2. Se adaugă un control de tip *Picture* pentru afişarea histogramei. Se modifică proprietățile acestuia (click dreapta şi *Properties*) (Fig. 3.4):
   a. Se modifică ID-ul acestuia în IDC_HISTOGRAM
   b. În secțiunea *Properties*, pentru aspect, *Client Edge* şi *Modal Frame* se pun pe valoarea *"True"*.

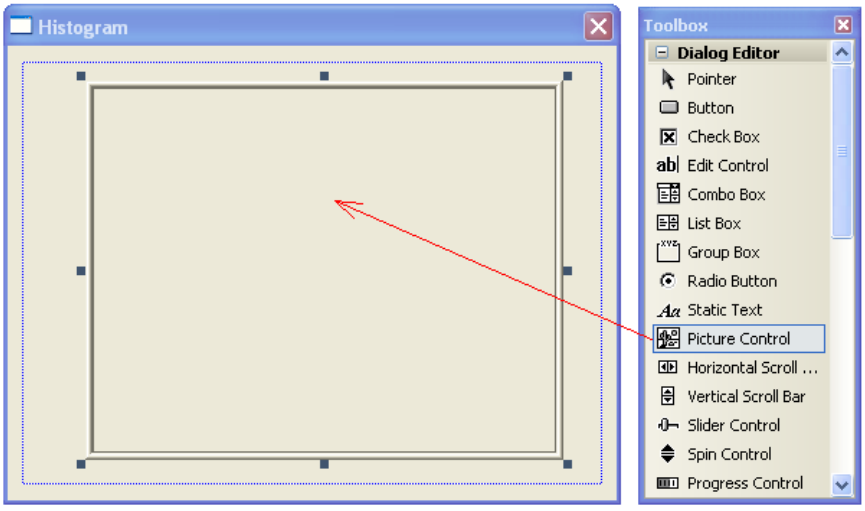


Fig. 3.4 Proiectarea căsuței de dialog pentru afişarea histogramei

3. Se creează o clasă nouă pentru *Dialog Box*-ul creat (click dreapta şi *Add Class…*). I se va da numele *CDlgHistogram* şi apoi se va închide *Wizard*-ul.
4. Se creează o clasă nouă pentru gestionarea controlului de afişare a histogramei. Se accesează meniul mediului de dezvoltare astfel (Fig. 3.5):
   a. *Project -> Add class…*
   b. Se alege categoria *Visual C++ -> MFC* şi apoi template-ul *MFC Class*
   c. Numele clasei va fi *CHistogram* având clasa de bază *CStatic*.

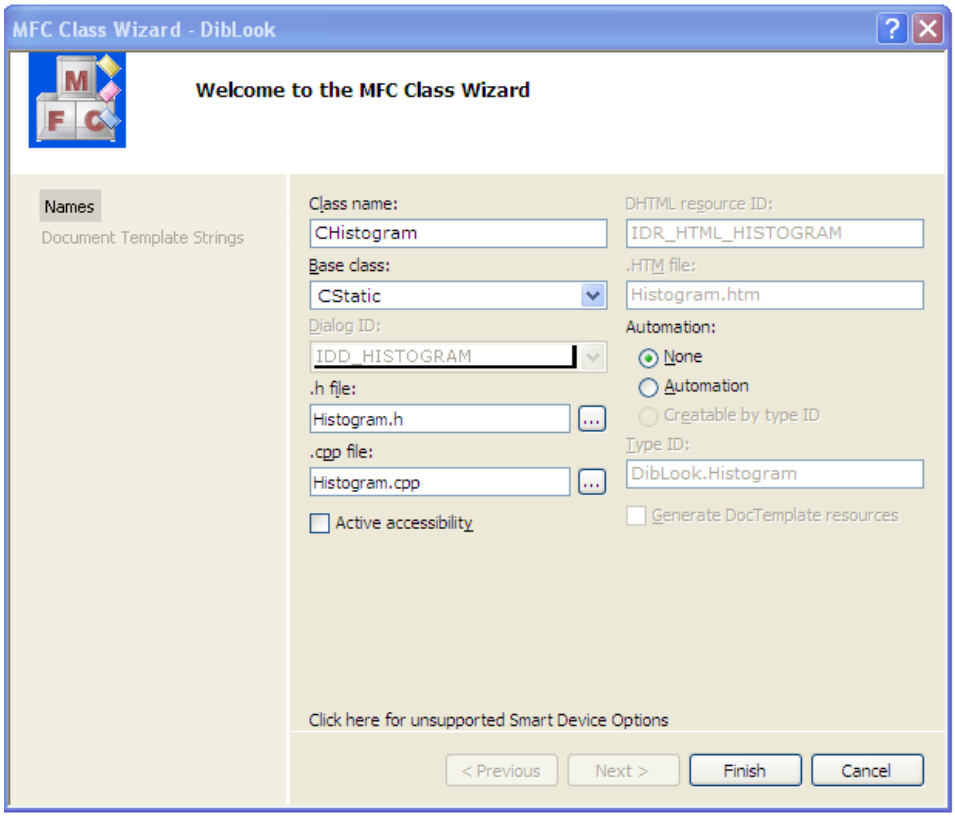


Fig. 3.5 Crearea clasei *CHistogram* pentru controlul de afişare a histogramei

5. În dialogul iniţial creat, pentru IDC_HISTOGRAM se ataşează variabila *m_Histogram* având categoria *Control* şi tipul *CHistogram* (Atenţie: trebuie ca în fişierul *CDlgHistogram.h* să se includă header-ul *Histogram.h*) (Fig. 3.6)

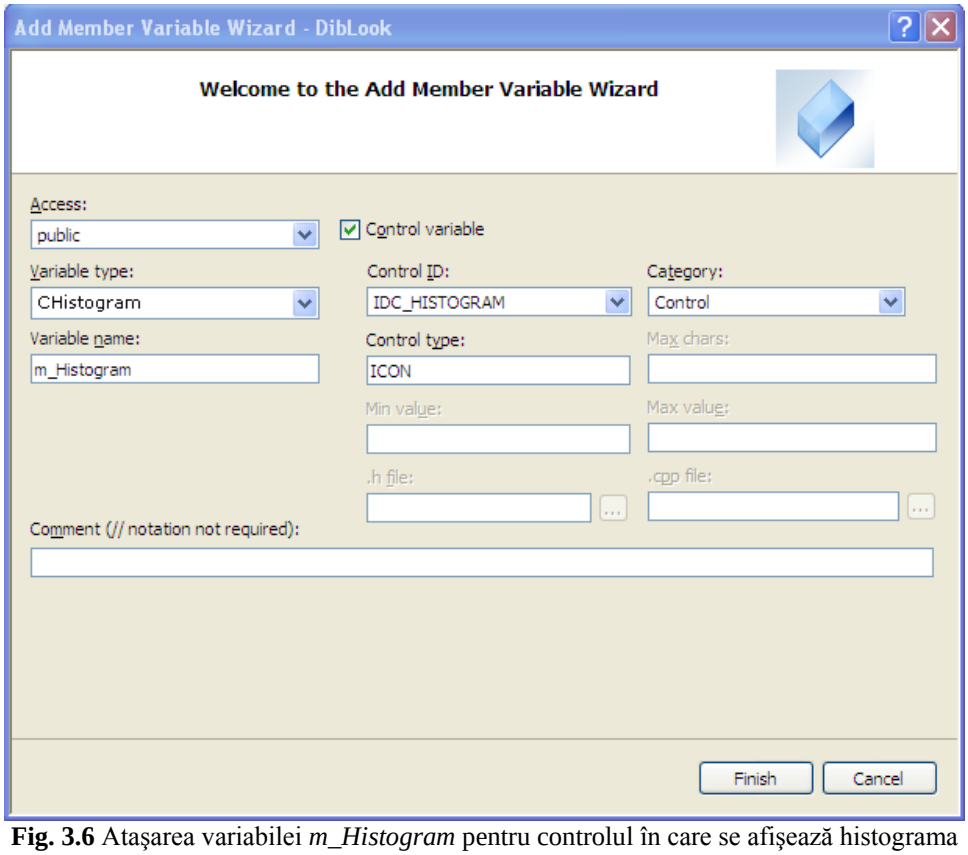


Fig. 3.6 Ataşarea variabilei *m_Histogram* pentru controlul în care se afişează histograma

6. Se atașează un handler pentru afișarea histogramei.
   a. În tabulatorul *Class View* click dreapta pe clasa *CHistogram* și apoi *Properties…*
   b. În fereastra *Properties* se alege secțiunea *Messages*
   c. La mesajul *WM_PAINT*, din lista derulantă se adaugă metoda *OnPaint* (*<Add> OnPaint*).

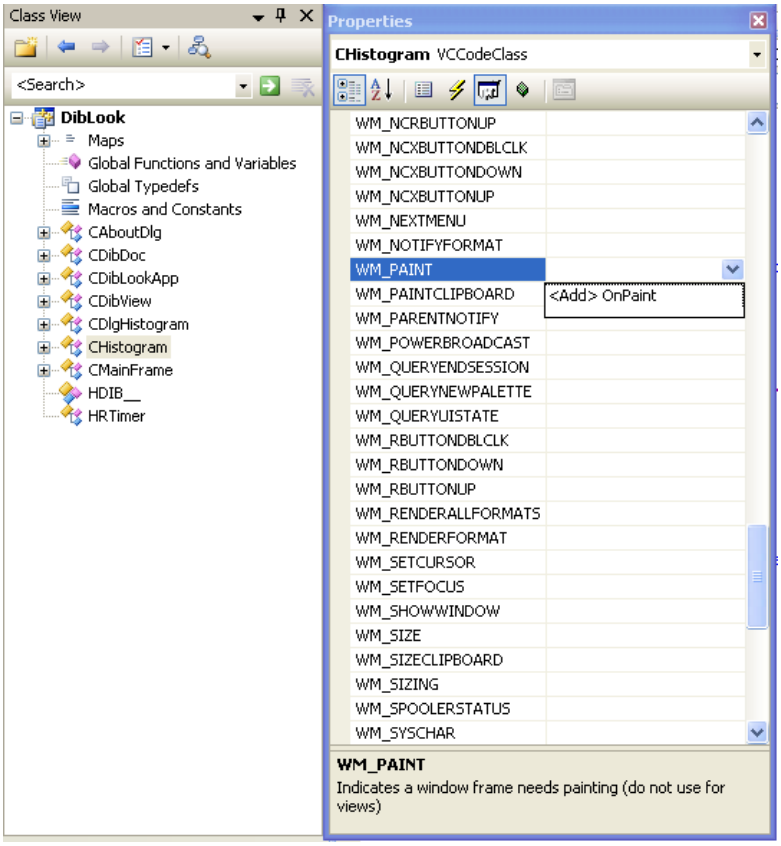


Fig. 3.7 Crearea metodei *OnPaint* atașată mesajului *WM_PAINT* clasei *CHistogram*

7. În header-ul fișierului *Histogram.h* la secțiunea *public* se va defini șirul de întregi *values[256]* reprezentând histograma care urmează a fi afișată (eventual scalată înainte)

```
class CHistogram : public CStatic
{
// Constructor
public:
    CHistogram();

// Atribute
public:
    int values[256];

// În liniile de mai jos clasa rămâne nemodificată
.
.
.
}
```

8. În fișierul sursă *Histogram.cpp*, metoda *OnPaint()* se va rescrie pentru afișarea (și eventual scalarea) histogramei definită de șirul de intrare *values[256]* astfel:

```
void CHistogram::OnPaint()
   {
       CPaintDC dc(this); // device context-ul pentru afişare
       CPen pen(PS_SOLID, 1, RGB(255,0,0)); // definirea pen-ului
                                            de desenare cu culoare roşie
       CPen *pTempPen=dc.SelectObject(&pen); // selectarea pen-ului de afişare
       CRect rect;
       GetClientRect(rect); // obţine zona de afişare dreptunghiulară disponibilă
       int height=rect.Height(); // înălţimea zonei de afişare
       int width=rect.Width(); // lăţimea zonei de afişare

       // se determină maximul şirului values[256]
       int i;
       int maxValue=0;
       for (i=0;i<256;i++)
            if (values[i]>maxValue)
                 maxValue=values[i];

       // se verifică dacă este nevoie de scalare
       double scaleFactor=1.0;
       if (maxValue>=height)
       {
            // este nevoie de scalare
            scaleFactor=(double)height/maxValue;
       }

       // se afişeaza histograma (eventual scalata) ca nişte bare verticale
       for (i=0;i<256;i++)
       {
            // determinarea lungimii liniei
            int lengthLine=(int)(scaleFactor*values[i]);
            //afişarea liniei
            dc.MoveTo(i,height);
            dc.LineTo(i,height-lengthLine);
       }

       dc.SelectObject(pTempPen); // restaurarea pen-ului de afişare
   }
```

9. Afişarea histogramei calculate în *Dialog-Box*-ul creat:
   a. Se adaugă un meniu de procesare pentru calculul histogramei şi o metodă asociată la apăsarea meniului respectiv, denumită *OnDisplayHistogram()*
   b. Se include fişierul *DlgHistogram.h* în fişierul *DibView.cpp*
   c. Metoda *OnDisplayHistogram()* ataşată evenimentului click pe meniul de procesare va trebui definită astfel:

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

     int histValues[256];
     float FDPValues[256];

     // se va scrie codul pentru calcularea histogramei în şirul de întregi
     histValues[256]
     // se va scrie codul pentru calcularea FDP în şirul de valori reale
     FDPValues[256]

     // se instanţiază un dialog box de afişare şi se asociază histograma
     CDlgHistogram dlg;
     memcpy(dlg.m_Histogram.values,histValues,sizeof(histValues));
     // se afişează dialog box-ul
     dlg.DoModal();

     END_SOURCE_PROCESSING;
}
```

3.4.2. Varianta 2: afişarea histogramei direct în zona utilizator a unei căsuțe de dialog

Se creează o fereastră de dialog nouă şi i se ataşează clasa *CDlgGrayStatistics*. Afişarea histogramei într-o fereastră de dialog ca şi cea prezentată în Fig. 3.8 se poate face prin definirea funcției *OnPaint* corespunzătoare mesajului *WM_PAINT* asociat căsuței/ferestrei de dialog (Fig. 3.9) ca şi în exemplul de mai jos:

```
#define LEFT 10
#define HEIGHT 100
#define BOTTOM 200

void CDlgGrayStatistics::OnPaint()
{
	CPaintDC dc(this); // device context for painting
	POINT pct;
	for (int g=0; g<256; g++) {
		pct.x=LEFT + g;
		int N=(float)(m_hist[g]*HEIGHT)/(float)m_maxhist;
		for (int n=0; n <N ; n++) {
			pct.y=BOTTOM-n;
			dc.SetPixel(pct,RGB(0,0,0));
		}
	}
}
```

Unde: `int *m_hist; int m_maxhist;` sunt variabile membre ale clasei dialog ce sunt inițializate în funcția de procesare.

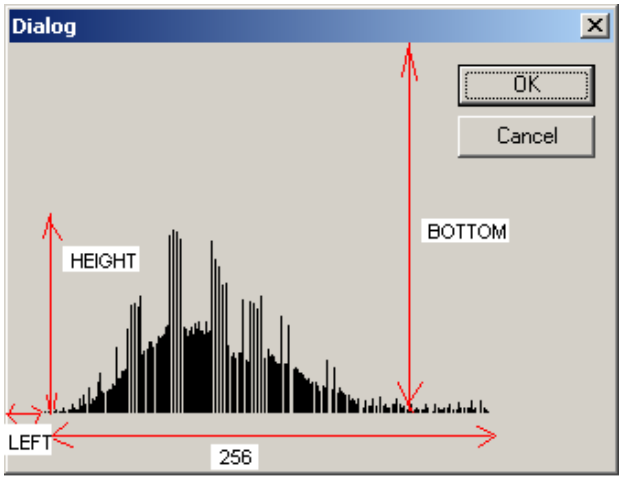


Fig. 3.8 Exemplu de afişarea histogramei într-o fereastră de dialog

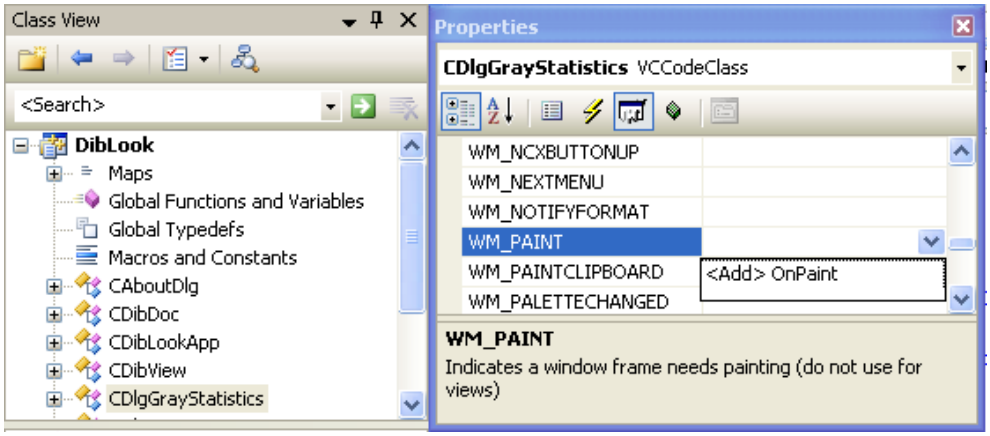


Fig. 3.9 Asocierea funcției *OnPaint* cu mesajul *WM_PAINT* de desenare a dialogului

3.5. Activități practice

1. Calculați histograma (într-un vector de întregi de dimensiune 256) şi FDP (într-un vector de tip float de dimensiune 256) pentru o imagine grayscale. Afișați histograma calculată utilizând una din metodele prezentate în secțiunea 3.4.
2. Implementați algoritmul de reducere a nivelurilor de gri (praguri multiple).
3. Îmbunătățiți algoritmul de reducere a nivelurilor de gri (praguri multiple) utilizând distribuirea erorii cu algoritmul Floyd-Steinberg.
4. **Salvați-vă ceea ce ați lucrat. Utilizați aceeaşi aplicație în laboratoarele viitoare. La sfârşitul laboratorului de procesare a imaginilor va trebui să prezentați propria aplicație cu algoritmii implementați!!!**

Referințe

[1]. R.C.Gonzales, R.E.Woods, *Digital Image Processing. 2-nd Edition*, Prentice Hall, 2002.
[2]. Algoritmul Floyd-Steinberg, http://en.wikipedia.org/wiki/Floyd-Steinberg_dithering