Region Interaction

Mriganka Haldavnekar

Department of ECE

Clemson University

# Introduction

This paper considers the concept of region interactions using GUI and event handling techniques. The software used for this GUI is Microsoft Visual Studio. The user can load the desired ppm image and perform region growing on it.

Region growing is the creation of regions in the given image on the basis of a number of criteria, also known as ‘predicates’. The pixels surrounding the seed pixel are checked for these predicates, and a decision is made whether they are to be added to the region or not. Here, we consider two predicates, namely- absolute difference of the pixel intensity of pixels already in region, and distance of a pixel to the centroid of the pixels already in the region.

Region growth is to be performed in two ways, viz. the step mode and the play mode, which will be discussed later.

# Methods

## Interface:

The interface.rc file in the project solution can be edited to give the desired menu options, and dialog boxes. The interface can be edited through the Visual Studio GUI, or by editing the code in interface.rc.

Dialog boxes have been created in order to allow the user enter the color desired for growing the region and the values for the predicates.

Some shortcuts have also been provided for faster access of the menu.

The code that has been used for this purpose can be found in the Appendix.

## Menu, modes and dialog boxes:

1. Load(l/L) allows the user to load a ppm image to the window for region growing.
2. Quit allows the user to quit the application.
3. Region grow:
   1. Play(p/P) allows the user to set a seed pixel, from which the region grows at the rate of 1ms.
   2. Step(s/S) allows the user to grow the region only when the ‘j’ key is pressed.

Toggling between Step and Play is possible.

1. Refresh(f/F) allows the user to clear the result of the previous region grow and load the original image again.
2. Predicates(d/D) opens a dialog box prompting the user to enter the values of the predicates, namely- absolute difference in intensity and distance from the centroid.
3. Color(c/C) opens a dialog box prompting the user to enter the RGB values for the desired color.
4. Shortcuts for incrementing the RGB values are r/R, g/G and b/B.
5. Pressing j/J allows the user to grow the region in the Step mode.

## Region growing

Region growing can be performed through calling the thread region ( ). The program code will check whether the Step or Play option has been selected through the menu IDs. The selection of either will be toggled between the two. If either of the option flags are set, the region ( ) thread will be begun and region will be grown.

1. In the Play mode, the process will be put to sleep using Sleep (1) i.e. sleep for 1ms after adding a pixel to the region and giving it the desired color.
2. In the Step mode, the region is grown only when the key ‘j’ is pressed and until Play is not pressed.

The predicate and the RGB values are initialized to 10, 30, 255, 0 and 0 respectively. Thereafter, that values the values read from the dialog box entries are read and passed to the region ( ) thread.

## Program code:

The program codes can be found in the Appendix:

Appendix A: resource.h

Appendix B: interface.rc

Appendix C: globals.h

Appendix D: main.c

# Results

## Interface, menu and dialog boxes:

The following are the Menus, and dialog boxes included in the GUI:

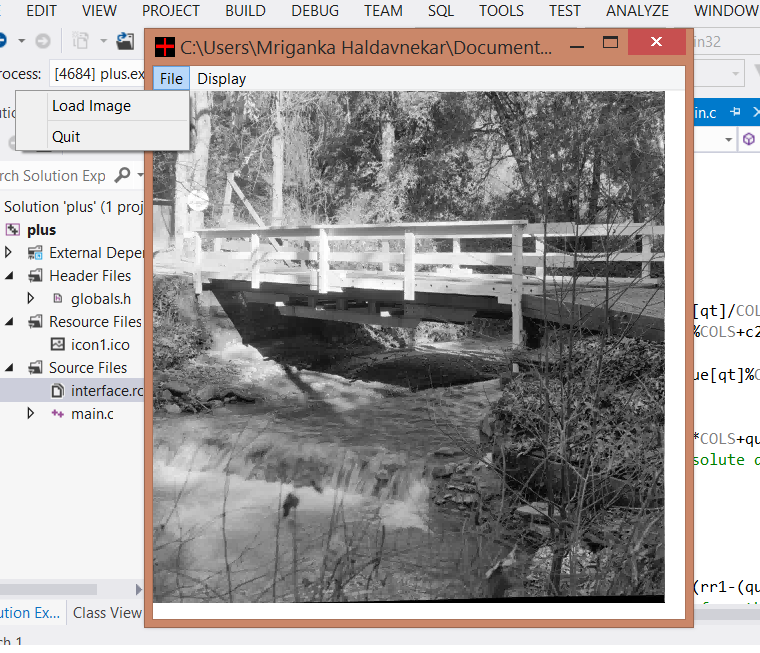


Figure 1 File menu

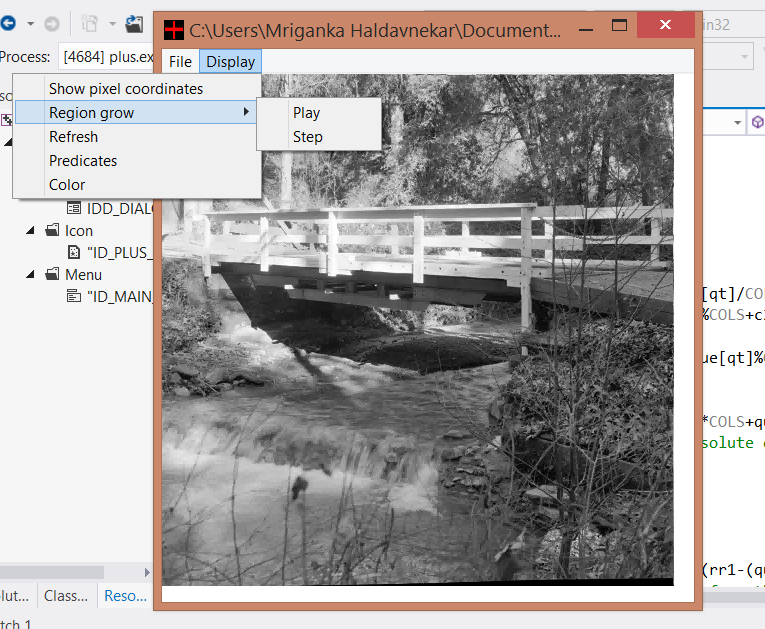


Figure 2 Display Menu

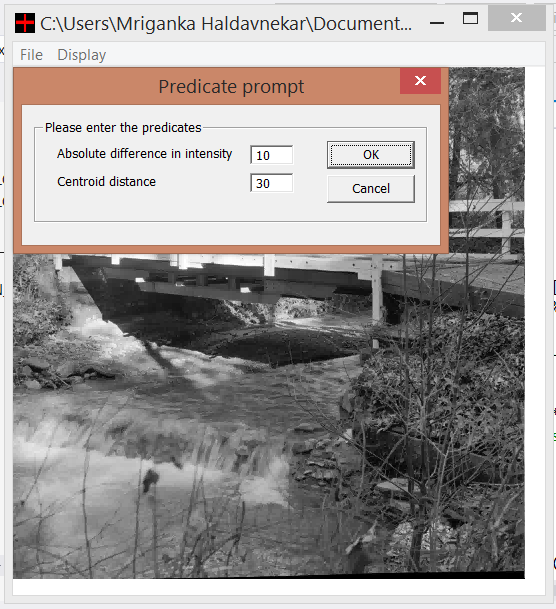


Figure 3 Predicate prompt

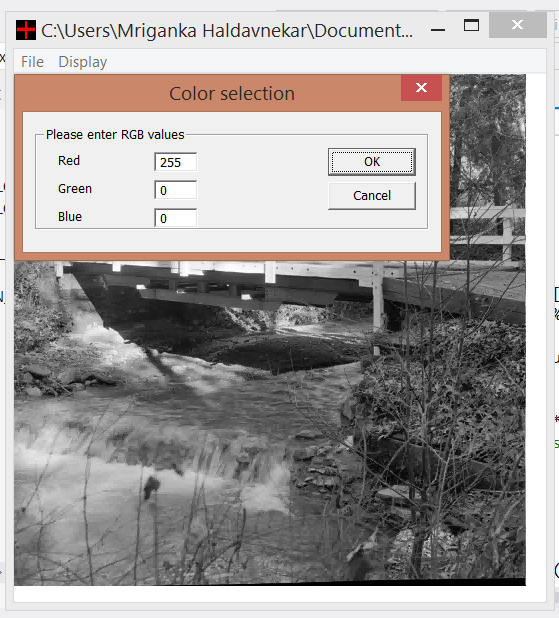


Figure 4 Color selection

## Region growth:

The following is an example of a grown region:



Figure 5 Region growth (absolute difference=10, distance from centroid=30)

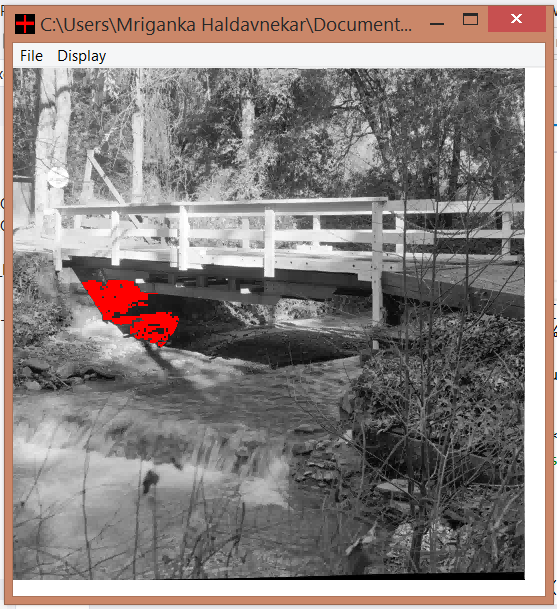


Figure 6 Region growth (absolute difference=10, distance from centroid=50)

# Conclusion

Thus, we have seen how to implement interactive region growing. We have observed the effect of change of predicates and color selection on the region growing.

Appendix

## Resource file

resource.h

//{{NO\_DEPENDENCIES}}

// Microsoft Visual C++ generated include file.

// Used by interface.rc

//

#define IDD\_DIALOG1 102

#define IDD\_DIALOG2 103

#define IDC\_EDIT1 1000

#define IDC\_EDIT2 1001

#define IDC\_EDIT3 1002

#define ID\_DISPLAY\_REGIONGROW 40001

#define ID\_REGIONGROW\_PLAY 40002

#define ID\_REGIONGROW\_STEP 40003

#define ID\_DISPLAY\_REFRESH 40004

#define ID\_DISPLAY\_PREDICATES 40005

#define ID\_DISPLAY\_COLOR 40006

#define ID\_FILE\_LOADI 40028

#define ID\_FILE\_LOAD 40028

#define ID\_FILE\_QUIT 40031

#define ID\_DISPLAY\_SHOWPIXELCOORDINATES 40042

#define ID\_SHOWPIXELCOORDS 40042

#define ID\_DISPLAY\_CLIPIMAGECOORDINATES 40043

#define ID\_DISPLAY\_PLOTSIGNAL 40047

#define ID\_LABEL\_HELP 40066

// Next default values for new objects

//

#ifdef APSTUDIO\_INVOKED

#ifndef APSTUDIO\_READONLY\_SYMBOLS

#define \_APS\_NEXT\_RESOURCE\_VALUE 104

#define \_APS\_NEXT\_COMMAND\_VALUE 40007

#define \_APS\_NEXT\_CONTROL\_VALUE 1001

#define \_APS\_NEXT\_SYMED\_VALUE 101

#endif

#endif

## Interface file

interface.rc

// Microsoft Visual C++ generated resource script.

//

#include "resource.h"

#define APSTUDIO\_READONLY\_SYMBOLS

/////////////////////////////////////////////////////////////////////////////

//

// Generated from the TEXTINCLUDE 2 resource.

//

#include "afxres.h"

/////////////////////////////////////////////////////////////////////////////

#undef APSTUDIO\_READONLY\_SYMBOLS

/////////////////////////////////////////////////////////////////////////////

// English (United States) resources

#if !defined(AFX\_RESOURCE\_DLL) || defined(AFX\_TARG\_ENU)

LANGUAGE LANG\_ENGLISH, SUBLANG\_ENGLISH\_US

#pragma code\_page(1252)

#ifdef APSTUDIO\_INVOKED

/////////////////////////////////////////////////////////////////////////////

//

// TEXTINCLUDE

//

1 TEXTINCLUDE

BEGIN

"resource.h\0"

END

2 TEXTINCLUDE

BEGIN

"#include ""afxres.h""\r\n"

"\0"

END

3 TEXTINCLUDE

BEGIN

"\r\n"

"\0"

END

#endif // APSTUDIO\_INVOKED

/////////////////////////////////////////////////////////////////////////////

//

// Menu

//

ID\_MAIN\_MENU MENU

BEGIN

POPUP "File"

BEGIN

MENUITEM "&Load Image", ID\_FILE\_LOAD

MENUITEM SEPARATOR

MENUITEM "&Quit", ID\_FILE\_QUIT

END

POPUP "Display"

BEGIN

MENUITEM "Show pixel coordinates", ID\_SHOWPIXELCOORDS

POPUP "Region grow"

BEGIN

MENUITEM "&Play", ID\_REGIONGROW\_PLAY

MENUITEM "&Step", ID\_REGIONGROW\_STEP

END

MENUITEM "Re&fresh", ID\_DISPLAY\_REFRESH

MENUITEM "Pre&dicates", ID\_DISPLAY\_PREDICATES

MENUITEM "&Color", ID\_DISPLAY\_COLOR

END

END

/////////////////////////////////////////////////////////////////////////////

//

// Icon

//

// Icon with lowest ID value placed first to ensure application icon

// remains consistent on all systems.

ID\_PLUS\_ICON ICON "icon1.ico"

/////////////////////////////////////////////////////////////////////////////

//

// Dialog

//

IDD\_DIALOG1 DIALOGEX 0, 0, 239, 70

STYLE DS\_SETFONT | DS\_MODALFRAME | DS\_FIXEDSYS | WS\_POPUP | WS\_CAPTION | WS\_SYSMENU

CAPTION "Predicate prompt"

FONT 8, "MS Shell Dlg", 400, 0, 0x1

BEGIN

DEFPUSHBUTTON "OK",IDOK,174,18,50,14

PUSHBUTTON "Cancel",IDCANCEL,174,35,50,14

LTEXT "Absolute difference in intensity",IDC\_STATIC,20,20,101,8

LTEXT "Centroid distance",IDC\_STATIC,20,34,58,8

GROUPBOX "Please enter the predicates",IDC\_STATIC,7,7,225,52

EDITTEXT IDC\_EDIT1,130,20,25,10,ES\_AUTOHSCROLL | ES\_NUMBER

EDITTEXT IDC\_EDIT2,130,34,25,10,ES\_AUTOHSCROLL | ES\_NUMBER

END

IDD\_DIALOG2 DIALOGEX 0, 0, 239, 70

STYLE DS\_SETFONT | DS\_MODALFRAME | DS\_FIXEDSYS | WS\_POPUP | WS\_CAPTION | WS\_SYSMENU

CAPTION "Color selection"

FONT 8, "MS Shell Dlg", 400, 0, 0x1

BEGIN

DEFPUSHBUTTON "OK",IDOK,174,18,50,14

PUSHBUTTON "Cancel",IDCANCEL,174,35,50,14

GROUPBOX "Please enter RGB values",IDC\_STATIC,7,7,225,52

LTEXT "Red",IDC\_STATIC,20,20,35,8

LTEXT "Green",IDC\_STATIC,20,34,35,8

LTEXT "Blue",IDC\_STATIC,20,48,35,8

EDITTEXT IDC\_EDIT1,75,20,25,10,ES\_AUTOHSCROLL | ES\_NUMBER

EDITTEXT IDC\_EDIT2,75,34,25,10,ES\_AUTOHSCROLL | ES\_NUMBER

EDITTEXT IDC\_EDIT3,75,48,25,10,ES\_AUTOHSCROLL | ES\_NUMBER

END

/////////////////////////////////////////////////////////////////////////////

//

// DESIGNINFO

//

#ifdef APSTUDIO\_INVOKED

GUIDELINES DESIGNINFO

BEGIN

IDD\_DIALOG1, DIALOG

BEGIN

LEFTMARGIN, 7

TOPMARGIN, 7

END

IDD\_DIALOG2, DIALOG

BEGIN

LEFTMARGIN, 7

TOPMARGIN, 7

END

END

#endif // APSTUDIO\_INVOKED

#endif // English (United States) resources

/////////////////////////////////////////////////////////////////////////////

#ifndef APSTUDIO\_INVOKED

/////////////////////////////////////////////////////////////////////////////

//

// Generated from the TEXTINCLUDE 3 resource.

//

/////////////////////////////////////////////////////////////////////////////

#endif // not APSTUDIO\_INVOKED

## Globals

global.h

#define SQR(x) ((x)\*(x)) /\* macro for square \*/

#ifndef M\_PI /\* in case M\_PI not found in math.h \*/

#define M\_PI 3.1415927

#endif

#ifndef M\_E

#define M\_E 2.718282

#endif

#define MAX\_FILENAME\_CHARS 320

char filename[MAX\_FILENAME\_CHARS];

HWND MainWnd;

// Display flags

int ShowPixelCoords;

int Play;

int Step;

int Refresh;

int Predicate;

int Color;

// Image data

unsigned char \*OriginalImage;

int ROWS,COLS;

int xp,yp;

int red;

int blue;

int green;

//Predicates

int absd;

int ctrd;

#define TIMER\_SECOND 1 /\* ID of timer used for animation \*/

// Drawing flags

int TimerRow,TimerCol;

int ThreadRow,ThreadCol;

int ThreadRunning;

int SetEnable;

// Function prototypes

LRESULT CALLBACK WndProc(HWND,UINT,WPARAM,LPARAM);

void PaintImage();

void AnimationThread(void \*); /\* passes address of window \*/

void region(void \*); /\* passes address of window \*/

void RegionGrow();

## Main function

main.c

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <time.h>

#include <sys/timeb.h>

#include <windows.h>

#include <wingdi.h>

#include <winuser.h>

#include <process.h> /\* needed for multithreading \*/

#include "resource.h"

#include "globals.h"

#define SQR(x) ((x)\*(x))

int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,

LPTSTR lpCmdLine, int nCmdShow)

{

MSG msg;

HWND hWnd;

WNDCLASS wc;

wc.style=CS\_HREDRAW | CS\_VREDRAW;

wc.lpfnWndProc=(WNDPROC)WndProc;

wc.cbClsExtra=0;

wc.cbWndExtra=0;

wc.hInstance=hInstance;

wc.hIcon=LoadIcon(hInstance,"ID\_PLUS\_ICON");

wc.hCursor=LoadCursor(NULL,IDC\_ARROW);

wc.hbrBackground=(HBRUSH)(COLOR\_WINDOW+1);

wc.lpszMenuName="ID\_MAIN\_MENU";

wc.lpszClassName="PLUS";

if (!RegisterClass(&wc))

return(FALSE);

hWnd=CreateWindow("PLUS","plus program",

WS\_OVERLAPPEDWINDOW | WS\_HSCROLL | WS\_VSCROLL,

CW\_USEDEFAULT,0,550,600,NULL,NULL,hInstance,NULL);

if (!hWnd)

return(FALSE);

ShowScrollBar(hWnd,SB\_BOTH,FALSE);

ShowWindow(hWnd,nCmdShow);

UpdateWindow(hWnd);

MainWnd=hWnd;

/\* initialising flags and variables \*/

ShowPixelCoords=0;

Play=0;

Step=0;

Refresh=0;

Predicate=0;

Color=0;

SetEnable=0;

red=255;

blue=0;

green=0;

absd=10;

ctrd=30;

strcpy(filename,"");

OriginalImage=NULL;

ROWS=COLS=0;

InvalidateRect(hWnd,NULL,TRUE);

UpdateWindow(hWnd);

while (GetMessage(&msg,NULL,0,0))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return(msg.wParam);

}

BOOL CALLBACK AboutDlgProc1(HWND hWnd, UINT Message, WPARAM wParam, LPARAM lParam)

{

switch(Message)

{

case WM\_INITDIALOG:

SetDlgItemInt(hWnd,IDC\_EDIT1,absd,FALSE);

SetDlgItemInt(hWnd,IDC\_EDIT2,ctrd,FALSE);

return TRUE;

case WM\_COMMAND:

switch(LOWORD(wParam))

{

BOOL bSuccess;

case IDOK:

absd=GetDlgItemInt(hWnd, IDC\_EDIT1, &bSuccess, FALSE);

ctrd=GetDlgItemInt(hWnd, IDC\_EDIT2, &bSuccess, FALSE);

EndDialog(hWnd, IDOK);

break;

case IDCANCEL:

EndDialog(hWnd, IDCANCEL);

break;

}

break;

default:

return FALSE;

}

return TRUE;

}

BOOL CALLBACK AboutDlgProc2(HWND hWnd, UINT Message, WPARAM wParam, LPARAM lParam)

{

switch(Message)

{

case WM\_INITDIALOG:

SetDlgItemInt(hWnd,IDC\_EDIT1,red,FALSE);

SetDlgItemInt(hWnd,IDC\_EDIT2,green,FALSE);

SetDlgItemInt(hWnd,IDC\_EDIT3,blue,FALSE);

return TRUE;

case WM\_COMMAND:

switch(LOWORD(wParam))

{

BOOL bSuccess;

case IDOK:

red=GetDlgItemInt(hWnd, IDC\_EDIT1, &bSuccess, FALSE);

blue=GetDlgItemInt(hWnd, IDC\_EDIT2, &bSuccess, FALSE);

green=GetDlgItemInt(hWnd, IDC\_EDIT3, &bSuccess, FALSE);

EndDialog(hWnd, IDOK);

break;

case IDCANCEL:

EndDialog(hWnd, IDCANCEL);

break;

}

break;

default:

return FALSE;

}

return TRUE;

}

LRESULT CALLBACK WndProc (HWND hWnd, UINT uMsg,

WPARAM wParam, LPARAM lParam)

{

HMENU hMenu;

OPENFILENAME ofn;

FILE \*fpt;

HDC hDC;

char header[320],text[320];

int BYTES,xPos,yPos;

switch (uMsg)

{

case WM\_COMMAND:

switch (LOWORD(wParam))

{

/\* set flags \*/

case ID\_SHOWPIXELCOORDS:

ShowPixelCoords=(ShowPixelCoords+1)%2;

PaintImage();

break;

case ID\_REGIONGROW\_PLAY:

Play=(!Play);

Step=0;

break;

case ID\_REGIONGROW\_STEP:

Step=(!Step);

Play=0;

break;

case ID\_DISPLAY\_PREDICATES:

Predicate=(!Predicate);

PaintImage();

break;

case ID\_DISPLAY\_COLOR:

Color=(!Color);

PaintImage();

break;

case ID\_DISPLAY\_REFRESH:

Refresh=(!Refresh);

ShowPixelCoords=0;

Play=0;

Step=0;

hMenu=GetMenu(MainWnd);

CheckMenuItem(hMenu,ID\_SHOWPIXELCOORDS,MF\_UNCHECKED);

CheckMenuItem(hMenu,ID\_DISPLAY\_REFRESH,MF\_UNCHECKED);

CheckMenuItem(hMenu,ID\_REGIONGROW\_PLAY,MF\_UNCHECKED);

CheckMenuItem(hMenu,ID\_REGIONGROW\_STEP,MF\_UNCHECKED);

CheckMenuItem(hMenu,ID\_SHOWPIXELCOORDS,MF\_UNCHECKED);

DrawMenuBar(hWnd);

PaintImage();

break;

case ID\_FILE\_LOAD:

if (OriginalImage != NULL)

{

free(OriginalImage);

OriginalImage=NULL;

}

memset(&(ofn),0,sizeof(ofn));

ofn.lStructSize=sizeof(ofn);

ofn.lpstrFile=filename;

filename[0]=0;

ofn.nMaxFile=MAX\_FILENAME\_CHARS;

ofn.Flags=OFN\_EXPLORER | OFN\_HIDEREADONLY;

ofn.lpstrFilter = "PPM files\0\*.ppm\0All files\0\*.\*\0\0";

if (!( GetOpenFileName(&ofn)) || filename[0] == '\0')

break; /\* user cancelled load \*/

if ((fpt=fopen(filename,"rb")) == NULL)

{

MessageBox(NULL,"Unable to open file",filename,MB\_OK | MB\_APPLMODAL);

break;

}

fscanf(fpt,"%s %d %d %d",header,&COLS,&ROWS,&BYTES);

if (strcmp(header,"P5") != 0 || BYTES != 255)

{

MessageBox(NULL,"Not a PPM (P5 greyscale) image",filename,MB\_OK | MB\_APPLMODAL);

fclose(fpt);

break;

}

OriginalImage=(unsigned char \*)calloc(ROWS\*COLS,1);

header[0]=fgetc(fpt); /\* whitespace character after header \*/

fread(OriginalImage,1,ROWS\*COLS,fpt);

fclose(fpt);

SetWindowText(hWnd,filename);

PaintImage();

break;

case ID\_FILE\_QUIT:

DestroyWindow(hWnd);

break;

}

break;

case WM\_SIZE: /\* could be used to detect when window size changes \*/

PaintImage();

return(DefWindowProc(hWnd,uMsg,wParam,lParam));

break;

case WM\_PAINT:

PaintImage();

return(DefWindowProc(hWnd,uMsg,wParam,lParam));

break;

case WM\_LBUTTONDOWN:

/\* selection of pixel to grow region from \*/

if (Play||Step == 1)

{

xPos=LOWORD(lParam);

yPos=HIWORD(lParam);

xp=xPos;

yp=yPos;

\_beginthread(region,0,MainWnd);

}

return(DefWindowProc(hWnd,uMsg,wParam,lParam));

break;

case WM\_MOUSEMOVE:

if (ShowPixelCoords == 1)

{

xPos=LOWORD(lParam);

yPos=HIWORD(lParam);

if (xPos >= 0 && xPos < COLS && yPos >= 0 && yPos < ROWS)

{

sprintf(text,"%d,%d=>%d ",xPos,yPos,OriginalImage[yPos\*COLS+xPos]);

hDC=GetDC(MainWnd);

TextOut(hDC,0,0,text,strlen(text)); /\* draw text on the window \*/

SetPixel(hDC,xPos,yPos,RGB(255,0,0)); /\* color the cursor position red \*/

ReleaseDC(MainWnd,hDC);

}

}

return(DefWindowProc(hWnd,uMsg,wParam,lParam));

break;

case WM\_KEYDOWN:

////////////////////////////\* shortcuts \*//////////////////////////////////

if (wParam == 'k' || wParam == 'K')

PostMessage(MainWnd,WM\_COMMAND,ID\_SHOWPIXELCOORDS,0); /\* show pixel co-ordinates \*/

if (wParam == 'p' || wParam == 'P')

PostMessage(MainWnd,WM\_COMMAND,ID\_REGIONGROW\_PLAY,0); /\* play mode \*/

if (wParam == 'f' || wParam == 'F')

PostMessage(MainWnd,WM\_COMMAND,ID\_DISPLAY\_REFRESH,0); /\* refresh \*/

if (wParam == 's' || wParam == 'S')

PostMessage(MainWnd,WM\_COMMAND,ID\_REGIONGROW\_STEP,0); /\* step \*/

if (wParam == 'c' || wParam == 'C')

PostMessage(MainWnd,WM\_COMMAND,ID\_DISPLAY\_COLOR,0); /\* color selection \*/

if (wParam == 'd' || wParam == 'D')

PostMessage(MainWnd,WM\_COMMAND,ID\_DISPLAY\_PREDICATES,0);/\* predicate selection \*/

if (wParam == 'l' || wParam == 'L')

PostMessage(MainWnd,WM\_COMMAND,ID\_FILE\_LOAD,0);/\* predicate selection \*/

if (wParam == 'j' || wParam == 'J')

SetEnable=1; /\* growing region in steps \*/

if (wParam == 'r' || wParam == 'R')

if((red+1)/255<=1)

red++; /\* increasing the pixel redness \*/

if (wParam == 'b' || wParam == 'B')

if((blue+1)/255<=1)

blue++; /\* increasing the pixel blueness \*/

if (wParam == 'g' || wParam == 'G')

if((green+1)/255<=1)

green++; /\* increasing the pixel greenness \*/

if (wParam == 'o' || wParam == 'O')

{

red=0;

blue=0;

green=0;

}

case WM\_TIMER: /\* this event gets triggered every time the timer goes off \*/

break;

case WM\_HSCROLL: /\* this event could be used to change what part of the image to draw \*/

PaintImage(); /\* direct PaintImage calls eliminate flicker; the alternative is InvalidateRect(hWnd,NULL,TRUE); UpdateWindow(hWnd); \*/

return(DefWindowProc(hWnd,uMsg,wParam,lParam));

break;

case WM\_VSCROLL: /\* this event could be used to change what part of the image to draw \*/

PaintImage();

return(DefWindowProc(hWnd,uMsg,wParam,lParam));

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return(DefWindowProc(hWnd,uMsg,wParam,lParam));

break;

}

hMenu=GetMenu(MainWnd);

if (ShowPixelCoords == 1)

CheckMenuItem(hMenu,ID\_SHOWPIXELCOORDS,MF\_CHECKED); /\* you can also call EnableMenuItem() to grey(disable) an option \*/

else

CheckMenuItem(hMenu,ID\_SHOWPIXELCOORDS,MF\_UNCHECKED);

if (Play == 1)

CheckMenuItem(hMenu,ID\_REGIONGROW\_PLAY,MF\_CHECKED); /\* you can also call EnableMenuItem() to grey(disable) an option \*/

else

CheckMenuItem(hMenu,ID\_REGIONGROW\_PLAY,MF\_UNCHECKED);

if (Step == 1)

CheckMenuItem(hMenu,ID\_REGIONGROW\_STEP,MF\_CHECKED); /\* you can also call EnableMenuItem() to grey(disable) an option \*/

else

CheckMenuItem(hMenu,ID\_REGIONGROW\_STEP,MF\_UNCHECKED);

if(Predicate==1)

{

DialogBox(GetModuleHandle(NULL), MAKEINTRESOURCE(IDD\_DIALOG1), hWnd, AboutDlgProc1);

Predicate=0;

}

if(Color==1)

{

DialogBox(GetModuleHandle(NULL), MAKEINTRESOURCE(IDD\_DIALOG2), hWnd, AboutDlgProc2);

Color=0;

}

DrawMenuBar(hWnd);

return(0L);

}

void PaintImage()

{

PAINTSTRUCT Painter;

HDC hDC;

BITMAPINFOHEADER bm\_info\_header;

BITMAPINFO \*bm\_info;

int i,r,c,DISPLAY\_ROWS,DISPLAY\_COLS;

unsigned char \*DisplayImage;

if (OriginalImage == NULL)

return; /\* no image to draw \*/

/\* Windows pads to 4-byte boundaries. We have to round the size up to 4 in each dimension, filling with black. \*/

DISPLAY\_ROWS=ROWS;

DISPLAY\_COLS=COLS;

if (DISPLAY\_ROWS % 4 != 0)

DISPLAY\_ROWS=(DISPLAY\_ROWS/4+1)\*4;

if (DISPLAY\_COLS % 4 != 0)

DISPLAY\_COLS=(DISPLAY\_COLS/4+1)\*4;

DisplayImage=(unsigned char \*)calloc(DISPLAY\_ROWS\*DISPLAY\_COLS,1);

for (r=0; r<ROWS; r++)

for (c=0; c<COLS; c++)

DisplayImage[r\*DISPLAY\_COLS+c]=OriginalImage[r\*COLS+c];

BeginPaint(MainWnd,&Painter);

hDC=GetDC(MainWnd);

bm\_info\_header.biSize=sizeof(BITMAPINFOHEADER);

bm\_info\_header.biWidth=DISPLAY\_COLS;

bm\_info\_header.biHeight=-DISPLAY\_ROWS;

bm\_info\_header.biPlanes=1;

bm\_info\_header.biBitCount=8;

bm\_info\_header.biCompression=BI\_RGB;

bm\_info\_header.biSizeImage=0;

bm\_info\_header.biXPelsPerMeter=0;

bm\_info\_header.biYPelsPerMeter=0;

bm\_info\_header.biClrUsed=256;

bm\_info\_header.biClrImportant=256;

bm\_info=(BITMAPINFO \*)calloc(1,sizeof(BITMAPINFO) + 256\*sizeof(RGBQUAD));

bm\_info->bmiHeader=bm\_info\_header;

for (i=0; i<256; i++)

{

bm\_info->bmiColors[i].rgbBlue=bm\_info->bmiColors[i].rgbGreen=bm\_info->bmiColors[i].rgbRed=i;

bm\_info->bmiColors[i].rgbReserved=0;

}

SetDIBitsToDevice(hDC,0,0,DISPLAY\_COLS,DISPLAY\_ROWS,0,0,

0, /\* first scan line \*/

DISPLAY\_ROWS, /\* number of scan lines \*/

DisplayImage,bm\_info,DIB\_RGB\_COLORS);

ReleaseDC(MainWnd,hDC);

EndPaint(MainWnd,&Painter);

free(DisplayImage);

free(bm\_info);

}

#define SQR(x) ((x)\*(x))

/\*

\*\* Given an image, a starting point, and a label, this routine

\*\* paint-fills (8-connected) the area with the given new label

\*\* according to the following criteria :

\*\* 1. absolute difference of the pixel intensity to the average

\*\* intensity of pixels alreadyin the region

\*\* 2. distance of the pixel to the centroidof the pixel already

\*\* in the region

\*/

void region(HWND AnimationWindowHandle)

{

HDC hDC;

unsigned char \*labels; /\* segmentation labels \*/

int r=yp;

int c=xp; /\* pixel to paint from \*/

int RegionSize;

labels=(unsigned char \*)calloc(ROWS\*COLS,sizeof(unsigned char));

hDC=GetDC(MainWnd);

RegionGrow(OriginalImage,labels,ROWS,COLS,r,c,0,255,&RegionSize,hDC);

ReleaseDC(MainWnd,hDC);

}

#define MAX\_QUEUE 10000 /\* max perimeter size (pixels) of border wavefront \*/

void RegionGrow(unsigned char \*image, /\* image data \*/

unsigned char \*labels, /\* segmentation labels \*/

int ROWS,int COLS, /\* size of image \*/

int r,int c, /\* pixel to paint from \*/

int paint\_over\_label, /\* image label to paint over \*/

int new\_label, /\* image label for painting \*/

int \*count,HDC hDC) /\* output: count of pixels painted \*/

{

int r2,c2;

int queue[MAX\_QUEUE],qh,qt;

int average,total; /\* average and total intensity in growing region \*/

/\* variables for centroid \*/

int rc=0;

int rr=0;

double rc1=0;

double rr1=0;

double area=0;

double d=0;

\*count=0;

if (labels[r\*COLS+c] != paint\_over\_label)

return;

labels[r\*COLS+c]=new\_label;

if(Refresh!=0)

{

SetPixel(hDC,c,r,RGB(red,green,blue));

}

average=total=(int)image[r\*COLS+c];

queue[0]=r\*COLS+c;

qh=1; /\* queue head \*/

qt=0; /\* queue tail \*/

(\*count)=1;

while (qt != qh)

{

if ((\*count)%50 == 0) /\* recalculate average after each 50 pixels join \*/

{

average=total/(\*count);

}

/\*compute centroid co-ordinate values \*/

rc1=(double)rc/area;

rr1=(double)rr/area;

/\* scan through 8 neighbours \*/

for (r2=-1; r2<=1; r2++)

for (c2=-1; c2<=1; c2++)

{

if (r2 == 0 && c2 == 0)

continue;

if ((queue[qt]/COLS+r2) < 0 || (queue[qt]/COLS+r2) >= ROWS ||

(queue[qt]%COLS+c2) < 0 || (queue[qt]%COLS+c2) >= COLS)

continue;

if (labels[(queue[qt]/COLS+r2)\*COLS+queue[qt]%COLS+c2]!=paint\_over\_label)

continue;

/\* test criteria to join region \*/

if (abs((int)(image[(queue[qt]/COLS+r2)\*COLS+queue[qt]%COLS+c2])

-average) > absd) /\* criterion for absolute difference in the intensities \*/

{

continue;

}

else

{

d=sqrt(SQR(rc1-(queue[qt]%COLS+c2))+SQR(rr1-(queue[qt]/COLS+r2)));

if (d > ctrd) /\* criterion for distance from the centroid of region \*/

continue;

}

labels[(queue[qt]/COLS+r2)\*COLS+queue[qt]%COLS+c2]=new\_label;

if(Play==1) /\* checking whether in play mode \*/

{

Sleep(1);/\* desired growth rate of 1ms\*/

SetPixel(hDC,queue[qt]%COLS+c2,queue[qt]/COLS+r2,RGB(red,green,blue));

}

if(Step==1) /\* checking whether in step mode \*/

{

while(SetEnable==0 && Play==0) /\* wait till 'j' is pressed and play is not enabled \*/

{

}

SetPixel(hDC,queue[qt]%COLS+c2,queue[qt]/COLS+r2,RGB(red,green,blue));

SetEnable=0;

}

total+=image[(queue[qt]/COLS+r2)\*COLS+queue[qt]%COLS+c2];

/\* computation for the centroid \*/

rc=rc+queue[qt]%COLS+c2;

rr=rr+queue[qt]/COLS+r2;

area++;

(\*count)++;

queue[qh]=(queue[qt]/COLS+r2)\*COLS+queue[qt]%COLS+c2;

qh=(qh+1)%MAX\_QUEUE;

if (qh == qt)

{

printf("Max queue size exceeded\n");

exit(0);

}

}

qt=(qt+1)%MAX\_QUEUE;

}

}