Set jednostavnih akcija nad slikom

Projekat iz računarske elektronike

Ilija Petković 0121-2014
Vuk Vukomanović 0018-2014
Modul OE
(Leto, 2017)

Elektrotehnički fakultet

19th June 2017

Contents

1 Postavka projekta

Napisati program koji sa učitava sliku u PGMA formatu i nakon učitavanja slike, sa standardnog ulaza učitava se akcija koju je potrebno izvršiti nad slikom. Naziv slike koja se učitava unosi se sa standardnog ulaza. Akcija koja se izvršava nad slikom definiše se unošenjem nekog od sledećih karaktera:

- **rl** Rotiranje slike za 90 °
- **rr** Rotiranje slike za -90 °
- **rf** Rotiranje slike za 180 °
- **mh** Horizontalno ogledanje slike
- **mf** Vertikalno ogledanje slike

2 Koncept

U ovom delu izveštaja je obješnjen pristup rešavanju problema obrade slike. Sve zadate operacije predstavljaju proste transformacije nad slikom, tj. matricom..

2.1 Rotiranje

Budući da je traženo rotiranje 90° a ne za proizvoljan ugao, dovoljno je samo proći kroz celu matricu i zameniti odgovarajuće indexe. Pretpostavimo da je slika A dimenzija NxM, tj. da ima N redova i M kolona, shodno tome elementi nove matrice B, invertovanih dimenzija MxN, se formiraju na sledeći način :

$$B[i][j] = A[j][M - i - 1] \tag{1}$$

Ovim je objašnjeno rotiranje za 90°. Za rotiranja od 180° i -90° je ili potrebno više puta koristiti objašnjeno rotiranje ili koristiti odgvarajuću zavisnost indexa.

2.2 Vertikalno ogledanje

Slično kao i sa rotiranjem korišćena je zavisnost indeksa piskela izlazne matrice i odgovarajućeg piksela izlazne matrice. U slucaju vertikalnog ogledanja :

$$B[i][j] = A[n - i - 1][j]; \tag{2}$$

2.3 Horizontalno ogledanje

Suprotan problem vertikalnom ogledanju, tj. ovaj put je potrebno izvršiti ogledanje po horizontalnoj osi:

$$B[i][j] = A[i][M - j - 1]; \tag{3}$$

3 Tok programa

Za procedure u kojima moze doći do greške postoji prost mehanizmam Exceptions. Exceptions je promneljiva koja sadrzi status uspeha procedure. Kod je podeljen u više logičkih celina. U procedurima je grupisan kod koji se potencijalno više puta koristi, od ostalih delova koda su napravljeni makroi radi bolje preglednosti koda. Kod je napisan u Irvine MASM asmebleru. Od ugradjenih funkcija iz irvione biblioteke su korićenje sledeće :

- **ReadString**
- **CreateOutputFile**
- **mWrite**
- **WriteToFile**
- **CloseFile**
- **WriteDec**
- **Crlf**

- **ParseDecimal32**

Procedure koje se nalaze u kodu su :

- **WriteToImage** Procedura za ispis buffera u file-a(ACII matricu)
- **ReadFromImage** Procedura za čitanje ASCII matrice(file-a) u buffer
- **Variable** Procedura za header-a slike
- **rotateLeft** Procedura za rotiranje „integer„ matrice za 90°
- **rotate180** Procedura za rotiranje „integer„ matrice za 180°
- **rotateRigth** Procedura za rotiranje „integer„ matrice za -90°
- **mirrorHor** Procedura za horiznotalno ogledanje „integer„ matrice
- **mirrorVer** Procedura za vertikalno ogledanje „integer„ matrice 90°

Pored napisanih procedura kod sadrži i nekoliko makro-a. U nastavku je objašnjen sam tok programa.
U main proceduri se prvo poziva **ReadFromImage** procedura koja u promenljivu **InputBuffer** upiše ceo ulazni fajl. Nakon toga se poziva **Variables** iz koje se dobijaju dimenzije slike i maksimalna vrednost piksela. Pošto ASCII matrica nije pogodna za dalju obradu ona se makroom kopira u „integer„ matricu. Integer matrica je matrica u kojoj svaki piksel ima jedan bajt za razliku od učitane matrice gde je svaka cifra piksela ASCII karakter pa samim tim jedan piksel sadrži od 1 do 3 bajtova. Nakon toga se poziva funkcija koju je korisnik uneo sa standardnog ulaza (rotate90, rotate180, rotate270, mirrorHor, mirrorVer). Izlaz procedure se smešta u pomoćni buffer. Nakon toga potrebno je prvo dodati header iz originalne slike (Ukoliko je pozvana rotateRight ili rotateLeft potrebno je invertovati dimenzije slike. Za kraj je potrebno pozvati makro za prebacivanje ovakve matrice u ASCII matricu sa dodatim headerom i proceduru **WriteToImage**.

3.1 Korisnički interfejs

Korisnički interfejs se sastoji od 3 dela. Pitanje za ime ulaznog fajla, željene operacije na učitanom slikom i ime izlaznog fajla.

4 Test primeri

Program je testiran na slikama koje su date sa postavom projekta. U nastavku je pokazan rad programa na slici balloons.pgm.

Nakon poziva procedure **RotateLeft**

Nakon poziva procedure **RotateRight**

Nakon poziva procedure **Rotate180**

Nakon poziva procedure **mirrorHor**

Nakon poziva procedure **mirrorVer**

Appendix A Code

```
;Project number 5 :
;Implements basic operations for image : rotate left, rotate right, mirror left,
     rotate 180 degrees , mirror right and mirror left
;GitHub repository :  https://github.com/vule12345/RE.git


INCLUDE Irvine32.inc
INCLUDE macros.inc

BUFFER_SIZE = 256*256*5
POM_SIZE = 50 * 50 * 4

.data
InputBuffer BYTE BUFFER_SIZE DUP(? )
TempBuffer BYTE BUFFER_SIZE DUP(? )
OutputBuffer BYTE BUFFER_SIZE DUP(? )
ImageBuffer BYTE BUFFER_SIZE DUP(? )
ProcedureHelper BYTE BUFFER_SIZE DUP(? )
```

```
ProcedureHelper1 BYTE BUFFER_SIZE DUP(? )
pomBuff BYTE POM_SIZE DUP(? )
addrese_First_M DWORD ?
size_M WORD 0
size_N WORD 0
counter_A DWORD 2
max_Value WORD ?
InputFilename BYTE 80 DUP(0)
Outfilename BYTE 80 DUP(0)
fileHandle  HANDLE ?
stringLength DWORD ?
i DWORD ?
j DWORD ?
N DWORD ?
M DWORD ?
NumOfPixels DWORD ?
Exception BYTE ?
stringIn DWORD ?

DigitHelper BYTE 3 DUP(? )
FinalImageSize DWORD ?
.code

 intToStrmacro macro
           Local petlja1, petlja2
 petlja1:
    mov ebx, 10
    xor edx, edx
    div ebx
    mov DigitHelper[esi], dl
    inc esi
    cmp eax, 0
    mov ebx,esi
    jz petlja2
    jmp petlja1

  petlja2:
    sub ebx,1
    mov dl, DigitHelper[ebx]
    add edx,30h
    mov byte ptr ImageBuffer[edi], dl
    inc edi
    cmp ebx,0
    jnz petlja2
 endm


main PROC
  ;Load ASCII image to InputBuffer
  call ReadFromImage
  ;Proc ReadfromImage raises Exception if there were any errors
  mov eax, 1
  cmp al, Exception
  jz quit

  mov  esi,OFFSET InputBuffer
  mov  ecx,SIZEOF InputBuffer
  mov  edi,OFFSET pomBuff

  push max_Value
  push addrese_First_M
    push size_M
```

```
  push size_N
push esi
push ecx
push edi
call Variables
add esp,12
pop size_N
pop size_M
pop addrese_First_M
pop max_Value
mov eax,0

;Move image width end heigth to N and M adn
;calculate ImageSize
xor eax, eax
mov ax, size_N
imul ax, size_M
sub eax, 1
mov [NumOfPixels] , eax
xor eax,eax
mov ax, size_N
mov N, eax
xor eax,eax
mov ax, size_M
mov M, eax

;Copy Input to BYTE matrix
cld
mov ecx,0
mov esi, 0
mov edx, addrese_First_M
mov edi, OFFSET TempBuffer
copy_image:

  cmp esi, NumOfPixels
  jz nastavak
  xor eax,eax
  mov al, byte ptr [edx+ecx]
  call IsDigit
  jnz maybeNewLine
  inc ecx
  jmp copy_image

maybeNewLine:
  push eax
  xor eax,eax
  mov al, byte ptr [edx+ecx+1]
  call IsDigit
  jz writeToOutput
  inc ecx
  jmp copy_image

writeToOutput:
  call ParseDecimal32
  ;call WriteDec
  mov [edi+esi], eax
  ;mWrite " "
  inc esi
  inc ecx
  add edx, ecx
  mov ecx, 0
  pop eax
```

```
  jmp copy_image

nastavak:
;Rest of proccessing

; push M
; push N
  ;push offset OutputBuffer
  ;push offset TempBuffer
  ;call mirrorVer
  ;add esp, 16

 ;User interface
  mWrite "Enter desired operation: "
  mov  edx,OFFSET stringIn
    mov  ecx,3
    call ReadString

mov eax, stringIn
 mov eax, 6C72h
 cmp eax, stringIn
 jnz nextOp
 ;Rotate left
  push NumOfPixels
  push M
  push N
  push offset OutputBuffer
  push offset TempBuffer
  call rotate90
  add esp, 20
  mov eax, N
  mov ebx, M
  mov N, ebx
  mov M, eax

  jmp PrintingToOutput

nextOp:
   ;rotate right
  mov eax, 7272h
  cmp eax, stringIN
  jnz nextOp1
  push NumOfPixels
  push M
  push N
  push offset OutputBuffer
  push offset TempBuffer
  call rotate270
  add esp, 20
  mov eax, N
  mov ebx, M
  mov N, ebx
  mov M, eax

  jmp PrintingToOutput

nextOp1:
;rotate 180

  mov eax, 6672h
  cmp eax, stringIN
  jnz nextOp2
```

```
  push NumOfPixels
  push M
  push N
  push offset OutputBuffer
  push offset TempBuffer
  call rotate180
  add esp, 20

  jmp PrintingToOutput

nextOp2:
;mirror horizontally
  mov eax, 686Dh
  cmp eax, stringIN
  jnz nextOp3
  push M
  push N
  push offset OutputBuffer
  push offset TempBuffer
  call mirrorHor
  add esp, 16

  jmp PrintingToOutput

nextOp3:
;mirror vertically
  mov eax, 766Dh
  cmp eax, stringIN
  jnz notAllowed
  push M
  push N
  push offset OutputBuffer
  push offset TempBuffer
  call mirrorVer
  add esp, 16


PrintingToOutput:
;Print header to output file
mov ecx, 0
mov edi, 0

;P2
mov byte ptr ImageBuffer[edi], 50h
inc edi
mov byte ptr ImageBuffer[edi], 32h
inc edi
mov byte ptr ImageBuffer[edi], 20h
inc edi

;Width
mov esi,0
mov eax, M
intToStrmacro
mov byte ptr ImageBuffer[edi], 20h
inc edi

;Height
mov esi,0
mov eax, N
intToStrmacro
mov byte ptr ImageBuffer[edi], 20h
```

```
  inc edi

  ;Maximum pixel value
  mov esi,0
  xor eax, eax
  mov ax, max_Value
  intToStrmacro

  ;Creating outputImage from matrix of ,,integeres,,
  ;Just allocating 1 byte for every digit and adding space after each number
  while1:
    mov byte ptr ImageBuffer[edi], 20h
    inc edi
    mov esi,0
    cmp ecx, NumOfPixels
    jz writings
    xor eax, eax
    mov al, byte ptr OutputBuffer[ecx]
    inc ecx
    intToStrmacro
    jmp while1
  writings:
  mov FinalImageSize, edi

  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  ;Call to WriteImage procedure
  ;Just writing final matrix to file
  call WriteToimage
  cmp al, Exception
  jz quit


  quit:
  exit
  notAllowed:
  mWrite "Operation not allowed"
main ENDP

WriteToImage PROC
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  ;Procedure for copying data from OutPutBuffer to output file
  ;Reading output file name for std
  mWrite "Enter name of output file: "
  mov edx, OFFSET Outfilename
  mov ecx, SIZEOF Outfilename
  call ReadString
  mov Exception, 0

  ;Creating new file with entered name
  mov edx, OFFSET Outfilename
  call CreateOutputFile
  mov fileHandle, eax

  ;Checking for errors
  cmp eax, INVALID_HANDLE_VALUE   ;Comparing if error is found
  jne out_file_ok           ; Jump to label out_file_ok which copies data
  mWrite <"Error occcurd while trying to open output file", 0dh, 0ah> ;Print error msg
       to std
  mov Exception,1
  jmp quit              ;Jump to end if error occurd
```

```
  ;Coping OutputBuffer to outut file
  out_file_ok:
  mov eax, fileHandle
  mov edx, OFFSET ImageBuffer
  mov ecx, FinalImageSize
  call WriteToFile          ;Writing to file
  mov eax, fileHandle
  call CloseFile            ;Closing file

  quit:
  ret
WriteToImage ENDP

ReadFromImage PROC
  ;Reading input file name from std input into input_file_name variable
  mWrite "Enter name of input file ?: "
  mov edx, OFFSET InputFilename
  mov ecx, SIZEOF InputFilename
  call ReadString

  mov Exception, 0
  ;Opening input file for reading data
  mov edx, OFFSET InputFilename
  call OpenInputFile
  mov fileHandle, eax

  ;Checking if there are eny mistakes
  cmp eax, INVALID_HANDLE_VALUE  ; Checking if there are mistakes
  jne input_file_ok     ;Jump to label ,,file_is_ok" if no exception were raised
  mWrite <"Can not open wanted file", 0dh, 0ah> ;Writing to std the error which occurd
  mov Exception, 1
  jmp quit            ;Jump to label ,,quit" , which is above exit

  ;Reading contets of file to buffer varriable
  input_file_ok:
    mov edx,OFFSET InputBuffer    ;Required parametars for ReadFromFile function
    mov ecx,BUFFER_SIZE     ;Required parametars for ReadFromFile function
    call ReadFromFile
    jnc check_buffer_size     ;Jump to label for checking if buffer size is enough
    mWrite "Error occurd while trying to read from input file."    ;Write reading
         error in windows box
    call WriteWindowsMsg
    mov Exception, 1
    jmp close_input_file      ;Jump to label for closing opened input file

  ;Checking if buffer is big enough
  check_buffer_size:
    cmp eax, BUFFER_SIZE      ;Buffer bigger than BUFFER_SIZE ?
    jb buf_size_ok          ;Jump to label for further proccesing
    mWrite <"Error occurd : buffer is too small ", 0dh, 0ah> ;Write error to std output
    mov Exception, 1
    jmp quit

  ;If buffer is ok
  buf_size_ok:
    mov InputBuffer[eax], 0     ;Inserting null terminator
    mWrite "File size :"       ;Displaying file size
    mov stringLength, eax
    call WriteDec
    call Crlf

  ;Closing input file
```

```
  close_input_file:
    mov eax, fileHandle
    call CloseFile
  quit:
    ret

ReadFromImage ENDP


rotate90 PROC c InputOffset:DWORD, OutputOffset:DWORD, N_size:DWORD, M_size:DWORD,
     img_size:DWORD
;Procedure for rotating image by 90 degrees
;param: InputOffset : Adress of input matrix
;param: OutputOffset : Adress of output matrix
;param row_size :  number of rows in image
;param img_size : complete number of bits
mov ecx , 0

  loopus:
    push ecx

    mov eax, ecx
    mov ebx, N_size
    xor edx, edx
    div ebx
    mov i, eax    ;eax =i/N
    mov j, edx    ;edx = j%N
    mul N_size; eax= i*N

    mov ebx, eax; ebx =i*N
    mov ecx, j ; ecx = j

    mov eax, j
    mul M_size ; eax = j*M
    mov edi, eax;  edi =j*M
    mov edx, M_size
    sub edx, i
    sub edx, 1; edx = M-i -1

    mov esi, InputOffset
    add esi, edi
    add esi, edx
    mov al, byte ptr [esi]

    mov esi, OutputOffset
    add esi,ebx
    add esi, ecx
    mov [esi], al
    pop ecx
    inc ecx
    cmp ecx, img_size
    jnz loopus

    ret

rotate90 ENDP

rotate180 PROC c InputOffset:DWORD, OutputOffset:DWORD, row_size:DWORD,
     column_size:DWORD,img_size:DWORD
;Procedure for rotating image by 180 degrees
;param: InputOffset : Adress of input matrix
;param: OutputOffset : Adress of output matrix
```

```
;param row_size :  number of rows in image
;param column_size :number of columns in image
;param img_size : complete number of bits

    push img_size
    push column_size
    push row_size
    push offset ProcedureHelper
    push InputOffset
    call rotate90
    add esp, 16

    push img_size
    push row_size
    push column_size
    push OutputOffset
    push offset ProcedureHelper
    call rotate90
    add esp, 16

    ret

rotate180 ENDP

rotate270 PROC c InputOffset:DWORD, OutputOffset:DWORD, row_size:DWORD,
     column_size:DWORD,img_size:DWORD
;Procedure for rotating image by 270 degrees
;param: InputOffset : Adress of input matrix
;param: OutputOffset : Adress of output matrix
;param row_size :  number of rows in image
;param column_size :number of columns in image
;param img_size : complete number of bits

    push img_size
    push column_size
    push row_size
    push offset ProcedureHelper
    push InputOffset
    call rotate90
    add esp, 16

    push img_size
    push row_size
    push column_size
    push offset ProcedureHelper1
    push offset ProcedureHelper
    call rotate90
    add esp, 16

    push img_size
    push column_size
    push row_size
    push OutputOffset
    push offset ProcedureHelper1
    call rotate90
    add esp, 16


    ret

rotate270 ENDP
```

```
mirrorHor  PROC c InputOffset:DWORD, OutputOffset:DWORD, row_size:DWORD,
     column_size:DWORD
;Procedure for horizontal mirror of image
;param: InputOffset : Adress of input matrix
;param: OutputOffset : Adress of output matrix
;param row_size :  number of rows in image
;param column_size :number of columns in image

xor ebx, ebx
xor ecx, ecx

    row_loop:
    mov ecx, 0
    column_loop:

    mov eax, ebx
    mul column_size ; eax=i*M
    mov edi, eax; edi=i*M

    mov edx,eax ; edx=i*M
    add edx, column_size
    sub edx, ecx
    sub edx, 1; edx=i*M+M-j-1

    mov esi, InputOffset
    add esi,edx
    mov al, [esi]
    add edi, ecx
    mov esi, OutputOffset
    add esi,edi
    mov [esi], al



    inc ecx
    cmp ecx, column_size
    jnz column_loop
    inc ebx
    cmp ebx, row_size
    jnz row_loop

    ret

mirrorHor ENDP

mirrorVer  PROC c InputOffset:DWORD, OutputOffset:DWORD, N_size:DWORD, M_size:DWORD
;Procedure for vertical mirror of image
;param: InputOffset : Adress of input matrix
;param: OutputOffset : Adress of output matrix
;param row_size :  number of rows in image
;param column_size :number of columns in image

xor ebx, ebx;i=0
xor ecx, ecx;j=0

    row_loop:
    mov ecx, 0
    column_loop:

    mov eax, ebx
    mul M_size ;eax=i*M
    add eax, ecx
```

```
    mov edi, eax ;edi =i*M+j
    add edi, OutputOffset

    mov eax, N_size
    sub eax,ebx
    sub eax,1  ;eax=N-i-1
    mul M_size ;
    add eax, ecx ;eax=(N-i-1)*M+j]
    mov esi, eax
    add esi, InputOffset

    mov al, [esi]
    mov [edi], al

    inc ecx
    cmp ecx, M_size
    jnz column_loop
    inc ebx
    cmp ebx, N_size
    jnz row_loop

    ret

mirrorVer ENDP

Variables PROC uses esi edi edx eax ebx
               LOCAL counter_m1:BYTE, degree1:WORD, size_M1:WORD, size_N1:WORD,
                    size_max:WORD

        mov size_M1,0
        mov size_N1,0
    mov counter_m1,0
    mov degree1,1
    mov size_max,0

    push ebp

    mov edi, [ebp+8]
        mov ecx, [ebp+12]
      mov esi, [ebp+16]

  to_Out_Buffer:
    lodsb
    stosb
    dec ecx
    cmp eax,'#'
    je next
    inc counter_m1
  next:
      lodsb
    stosb
    cmp eax,0ah
    je m_On
    loop next
    m_On:
    cmp counter_m1,2
    je picture_Size
    cmp counter_m1,3
    je remember
    loop to_Out_Buffer

    remember:
```

```
    mov degree1,1
  push eax
  push esi
  push ebx
  mov esi, 2
  mov ebx, 10
loop2:
    mov edx,edi
  sub edx,esi
  mov eax,0
  mov al,[edx]

  cmp al,0ah
  jz max_value_end

  sub ax,30h
  mul degree1
  add size_max,ax
  mov ax,degree1
  mul bx
  mov degree1,ax
  inc esi

  jmp loop2



  max_value_end:
    pop ebx
  pop esi
  pop eax
    mov ax,size_N1
  mov [ebp+20],ax
  mov ax,size_M1
  mov [ebp+22],ax
  mov eax,esi
  mov [ebp+24],eax
  mov ax,size_max
  mov [ebp+28],ax
  jmp kraj

  ;;;;

  picture_Size:
  push eax
  push esi
  push ebx
  mov esi, 2
  mov ebx, 10
loop1:
    mov edx,edi
  sub edx,esi
  mov eax,0
  mov al,[edx]

  cmp al,20h
  jz jump1

  sub ax,30h
  mul degree1
  add size_N1,ax
  mov ax,degree1
```

```
    mul bx
    mov degree1,ax
    inc esi

    jmp loop1

    jump1:
    mov degree1,1
      jmp picture_Size1

  picture_Size1:

      inc esi
      mov edx,edi
    sub edx,esi
    mov eax,0
    mov al,[edx]

    cmp eax,0ah
    jz jump

    sub ax,30h
    mul degree1
    add size_M1,ax
    mov ax,degree1
    mul bx
    mov degree1,ax

    jmp picture_Size1

  jump:
    pop ebx
    pop esi
    pop eax
    jmp to_Out_Buffer

  kraj:
      pop ebp
    ret


 Variables ENDP

 intToStrmacro macro
 mov eax,0
 endm
END main
```

References

[1] Assembly Language for X86 Processors, Kip Irvine

[2] Prezentacije sa vežbi