Painter

Muhammed Yahya 144082

Mukhtar Abdraheem 144090

Muhammed Nagdi 143119

Kahlid Nabigh 144029

Mohammed Salah 144074

|Micro-processor & Assembly language |

17-4-2018

\*Topics of the report

1-Brief Document About The project

2-Objective and the Program

3-what did you learn

4-the algorithm used and the pseudo-code

5-The Task Achieved and the problems

1-Breif Introduction about the project.

The project is to develop a painter that allows the user to paint in a white backgrounded painter with different colors selected by the user form our graphical user interface and the colors table contains various colors ranged from the black, yellow, gray, red,…etc. Also the user have the ability to use the keyboard to choose one of our geometric shapes such as line, circle, triangle and square.

The project is developed to run on 16 bits processor and was developed using the assembly programming language and the interrupts.

The I/O devices used in the this project are the keyboard-which is initialized by default to the processor and the mouse which have another story that needs to write processes to the mouse the first step is to make the mouse ready to communicate with the port and takes the instruction and the second write process is to tell the mouse to operate and allows us to read the status of the mouse and the x, y and z axis data.

2-The objective

We want to develop a graphical user interface painters that has our own logo.

The painter allows the user to draw the free diagram using the existing colors in our colors table. The colors can be chosen from the colors table by the GUI interface.

The painter allows the user to use the keyboard to choose whatever geometric shape available in our Painter, we have line, triangle, square and circle.

3-What did you learn?

We have learn quit a lots of assembly programming stuffs and interrupts but the most important things we have learned are:

1-To develop and Write a Pseudo-Code and to put all the possible cases in this code and then hurry up writing the assembly code which really saved as considerable amount of time comparing to our first days with project when we were not pay attention to the importance and the necessity of the Pseudo-Code.

2-The effectiveness and the importance of the Team Work to achieve the desired results

3-how to increase the number of code Bytes far more than the default 512 bytes allowed by the 16 bit processor programs and OS

4-sending commands to mouse (to port 0x60). That must be preceded by sending a 0xd4 byte to port 0x64 after waiting for bit 1 in 0x64 to be zero that indicates the mouse is ready to get command. We used 0xf4 (enable command)

5-how to determine whether the received data came from the moue or the keyboard

(Note: the mouse and the keyboard sends the data in the same byte way but the difference is in the bit 5(bit number 6) 0x20-so you can decide the type of the device sending the interrupts via the 0x64 port by Anding the byte received with 0x20 if the result is zero then the data are send via the keyboard device else the data belongs to the mouse.

6-Make Boarders for the painter so the painter allows the drawing within the specified Margins-Boards – and the drawings are no longer random one

7-when you want to do any arithmetic operation you must use the floating point unit since both the multiplication and the division is done on the ax register and the graphic interface and interrupts use the ax register mainly so if by accident the value encountered by the division or multiplication is the same as the value of any interrupt you will probably encounter some problems with the desired values.

8-Beside the text mode the monitor has graphics mode, which allow us to divide the screen into 320 x 200 pixels, we do that by using interrupt( int-10h).

9-using interrupts to communicate with I/O devices is easier than polling their registers.

10-we can increase the size of the program by use interrupts int 13h, and but ah equals to 0x02 and make al equals the number of the desired sectors such as

mov ah , 0x02

mov al ,6

mov dl , 0x80

mov ch ,0

mov dh , 0

mov cl , 2

mov bx , StartingTheCode

int 0x13

11-we know the instruction RDSTC which read the number of cycles of the processor from the last reset of the processor and used number of cycles read to generate the random number

5- The algorithms and the Pseudo-Code

The geometric shapes have algorithms and Pseudo-Code But the mouse, keyboard, the logo, colors Table and the overall Program have Pseudo-Code.

The Geometric shapes:

1. The line :

This sweet Part of the Program represent the Back-Bone For the others geometric shapes except the circle. At the developing phase. We have used three different method to draw the line each one have its advantages and Disadvantages and we have finally made up our mind to use the DDA line algorithm in computer science.

The other two algorithms are the line equation algorithm in which by having the first and the last points we can get the line equation in the following general form

Y=slope\*x + constant

-where the slope is (y2-y1)/(x2-x1)

-constant equal ( y1-slope\*x1)

The main problem with this algorithm is when the first and the next point have the same x axis values and so the points varies only on the y axis here the slope is infinite and the painter will draw only the first and the last point and on the board

The pseudo code for this Method is

1-find the slope using the above slope equation.

2-find the constant term.

3-use the above to values to find the general equation.

4-loop for the value of x from x1 until x2.

2-The modified Line algorithms Used:

We developed to line algorithms to solve the problem of the infinite or large slope before we used the DDA line algorithm, and those two modified line algorithms are as follow:

i-the first one says so : “if the slope is relative large –we make five as the threshold possible slope –then take the next value of the y axis and the first value of the y axis and the second value of the x axis ,fix the value of the x axis and increase the values of the first y axis until reach the second value of the y axis”

note we make sure that the next y axis is larger than the first one if not then swap them.

This code also generates lines starts from the top to the bottom of the screen in the case of the two points have the same x axis “we do not know the why???!!!!!”, but also solve the problem of the finite and large slope.

The Pseudo-Code:

1-find the slope of the line.

2-find the constant.

3-develop the line equation.

4-determine the larger y axis point and swap the values if y1 is larger than y2.

5-loop for the values of y –starting from y1 until y2 –while make x2 fixed, and fill all the values of y with x2 constant.

ii-The second modified method used is depend main on deceive the eye of the user for the two points which have x axis value with separation less than 8 pixels -2.5% of the x axis 320\*200 pixels- by trial and error we found that the most of the large value of the slope are for the two points with

1.2: The DDA line Algorithm:

// calculate dx , dy

dx = X1 - X0;

dy = Y1 - Y0;

// Depending upon absolute value of dx & dy

// choose number of steps to put pixel as

// steps = abs(dx) > abs(dy) ? abs(dx) : abs(dy)

steps = abs(dx) > abs(dy) ? abs(dx0 : abs(dy);

// calculate increment in x & y for each steps

Xinc = dx / (float) steps;

Yinc = dy / (float) steps;

// Put pixel for each step

X = X0;

Y = Y0;

for (int i = 0; i <= steps; i++)

{

putpixel (X,Y,color);

X += Xinc;

Y += Yinc;

}

The pseudo-Code is as follow:

1-find the absolute value of the delta x.

2-find the absolute value of delta y.

3- find the maximum of the absolute of the two deltas.

4-the maximum delta is the step and it determine the number of points in the loop with respect to x and y whatever one of them is the maximum

5-for the other variable with the minimum delta we make it increase the position to draw in by deltasmall/deltalarge assume deltax is less than deltay then we make x increased each time by the total sum of n\*deltax/deltay “using the rounding function” note the n variable represent the value of the loop counter

2-The Circle:

The pseudo-code for the circle

1-determine the center and the radius of the circle

2-then use the midpoint algorithm to draw the circle

3-the midpoint algorithm works as follow

3.1-after decide the center and the radius program decide 8 points as follow

(x,y),(-x,y),(x,-y),(-x,-y),(y,x),(-y,x),(y,-x),(-y,-x).

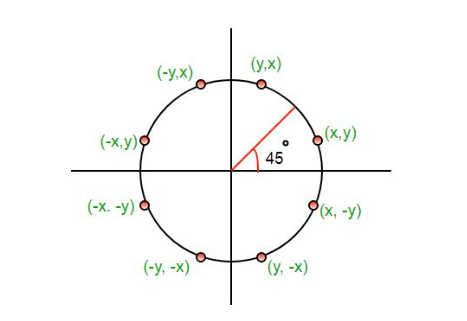
3.2-the algorithm allows you to increase the value of x and y depend on the value of the variable P

If P>0

1-then let (x-1,y+1) and calculate the new p

Else the new point (x,y+1) and calculate a new value of p by another equation,

4- we stop when we complete the first 1/8 of the circle



// C program for implementing

// Mid-Point Circle Drawing Algorithm

#include<stdio.h>

// Implementing Mid-Point Circle Drawing Algorithm

void midPointCircleDraw(int x\_centre, int y\_centre, int r)

{

    int x = r, y = 0;

    // Printing the initial point on the axes

    // after translation

    printf("(%d, %d) ", x + x\_centre, y + y\_centre);

    // When radius is zero only a single

    // point will be printed

    if (r > 0)

    {

        printf("(%d, %d) ", x + x\_centre, -y + y\_centre);

        printf("(%d, %d) ", y + x\_centre, x + y\_centre);

        printf("(%d, %d)\n", -y + x\_centre, x + y\_centre);

    }

    // Initialising the value of P

    int P = 1 - r;

    while (x > y)

    {

        y++;

        // Mid-point is inside or on the perimeter

        if (P <= 0)

            P = P + 2\*y + 1;

        // Mid-point is outside the perimeter

        else

        {

            x--;

            P = P + 2\*y - 2\*x + 1;

        }

        // All the perimeter points have already been printed

        if (x < y)

            break;

        // Printing the generated point and its reflection

        // in the other octants after translation

        printf("(%d, %d) ", x + x\_centre, y + y\_centre);

        printf("(%d, %d) ", -x + x\_centre, y + y\_centre);

        printf("(%d, %d) ", x + x\_centre, -y + y\_centre);

        printf("(%d, %d)\n", -x + x\_centre, -y + y\_centre);

        // If the generated point is on the line x = y then

        // the perimeter points have already been printed

        if (x != y)

        {

            printf("(%d, %d) ", y + x\_centre, x + y\_centre);

            printf("(%d, %d) ", -y + x\_centre, x + y\_centre);

            printf("(%d, %d) ", y + x\_centre, -x + y\_centre);

            printf("(%d, %d)\n", -y + x\_centre, -x + y\_centre);

        }

    }

}

// Driver code

int main()

{

    // To draw a circle of radius 3 centred at (0, 0)

    midPointCircleDraw(0, 0, 3);

    return 0;

}

3-The triangle:

The triangle code is based on three pressed from the mouse and calls the function paint three time the first time its start from the point (x0,y0) to

(x1,y1),then from the (x1,y1) to the point (x2,y2), and finally from the point(x2,y2) to (x0,y0)

Pseudo-Code:

1-press the mouse three times each to the corresponding triangle head

2-draw line from the point (x0,y0) to

(x1,y1),

3- From the (x1,y1) to the point (x2,y2),

4- from the point(x2,y2) to (x0,y0)

$

4-the Square:

The square is based on just two pressed to verify the length of the branch and then draw the line form the two inserted points and the line from the point (X0,y0) to the point (x0,y0+branchlength),and then draw line from the point(x0,yo+branchlength) to the point (x1,y1+branchlength),and then draw line from the point (x1,y1+branchlength) to the point(x1,y1)

Note :our square is to the bottom by default

Pseudo-Code:

1-select two line of the square

2-calculate the length of the branch of the square

3-draw line from the two selected points(x0,y0) to (x1,y1)

4-drw line from the point (X0,y0) to the point (x0,y0+branchlength),

5-draw line from the point(x0,yo+branchlength) to the point (x1,y1+branchlength)

6-draw line from the point (x1,y1+branchlength) to the point(x1,y1)

Now after we have windup with geometric shape we give the pseudo code and the flowcharts of the other part of the program.

1-Logo

The logo is based on draw the four different colors after each other with a specific amount of time known as long\_lag and the long\_lag is defined to be 20,000,000 loop counter which have 9 nop operation on the loop –the nop operation is which take time realy-after the four colors are drawn on the screen the logo is held for amount of time equals to verylong\_lag which is

100,000,000 loop with 9 nop operation and then the logo vanished and the sweet white background comes back to life by calling a specific part of code-the part that is responsible of drawing the white background from the very beginging

The pseudo-Code of the logo is as follows :

1-fill all the screen by white color

2-fill the screen from the pixels 110,210 on the x axis and from the point 50,80 on the yaxis with the blue color and then call the function lag

3-fill the screen from the pixels210 to 180 on the x axis and the points 80 to 150

On the y axis with the magnda color and then call the function lag

4-fill the screen from the pixels 110 to 180 on the axis and 150 to 120 on the y axis with the red color and then call the function lag

5-fill the screen from pixel of 110 to 140 on the x axis and the point from the 120 to 50 on the y axis with the gyan color and then call the function lag

6-after the logo has been drawn then call function lag-very-long

2-The Colors Table.

The idea of the colors table is color a specified pixels with the desired color and make the cursor colors each block by its color so the color table is not distorted by the cursor color “for example if the cursor color is yellow and you want t0 choose the red color which is at the top left corner ,the cursor will go through at least the black color block and then at the red color block ; if the cursor is not to be set to color each block by its color then the color of each block will change and distorted.

The pseudo code for the Color Table is as follow:

1-color each specified pixels block with the desired color

Note: you already have an idea of each block

2-check whether the left mouse button is clicked or not

3-if the left button is clicked then change the cursor color to the corresponding block in the block table so you donot distort the color of the block by the color of the cursor

3-The mouse code.

The mouse code is based on the interrupts and the assembly

To use the mouse first you have to clear the interrupt using the command cli, and then initialize the graphics mode by the commands

Mov ax,13

Int 10h

And then write to the mouse the command 0xd4 to make it ready to receive the command –reset -and after make sure there is a device connected and then write 0xf4 to enable the device , after this point you donot know whether the connected device is mouse so and the value written from the board 0x64 with 0x20 and if the result is zero then the data comes from the keyboard else from the mouse.

We have four reads that we must do and those are :

1. The first one is to read the status of the mouse whether there is button pushed or not
2. The second one is to read the status of the x axis
3. The third read is to define the status of the y axis
4. The fourth one is to read the value of the z axis

Pseudo-Code:

1-clear the interrupts

2-initilize the graphics mode

3-reset the mouse and that by write the value 0xd4 to the port 0x64

4-enable the mouse and that by write the value 0xf4 to the mouse

5-Anding the value read from the port 0x64 with 0x20 and if the result is zero the data comes from the keyboard else it came from the mouse

6-after make sure the data are from the mouse then you must encounter with four reads operations

7-the first read operation enables you to know the status of the mouse and if there is button pressed or not and this is done by and the value read from the port 0x60 –note before we reached to this point we have made sure that the data came from the general port 0x64 are came from the mouse-and after the anding process if the result in the al is zero then there is no key pressed ,if one the left button is pressed ,else the right button is pressed

8-the next read process from the port 0x60-note the read process is done by the command in al, 0x60-enables you to know the dislodges from the previous x axis position

9-the next read process from the port 0x60 enables you to know the z axis verifications

/////////////the end of the mouse operation

4-The Erase Operation

The erase command is done by press the button E in the keyboard and after the button is read from the keyboard the background is coloured with the white color form the point zero, zero until the boarder of the color table

5-The Spray(Bakhakh)

**Bakhakh Algorithm**

* **random function:**

firstly we used an “ read stamp-counter” (rdstc) instruction that put an 64 bit random number of the clock-cycles since the last reset of the processor

the instruction reserves its value and divide it into the registers EAX and EDX

and we are using the EAX register for the multiplication and division operations

the EDX was set to zero to accept the modulus of the division

the border of the square that the bakhakh is limited to a value of 5

ebx was set to 5

then the random number of cycles is divided to the value of 5

and the random number in edx is saved in [random1].

* **painter function:**

takes the color to be used by the interrupt from memory [color3]

and sets the coordinates of from [xrandom ] and [yrandom].

* **checker function:**

sets the borders of xrandom and yrandom is that 0,259,0,199 respectively for x and y.

* **Draw\_bakhakh function:**

It firstly takes the current coordinates from the mouse and implies it as a center for the operation ; the operation is built on 4 loops that represent the number of quarters in which the spots are to be drawn

For each loop (quarter) 4 spots are drawn countered by the register ESI

The nature of quarters:

Its primary reason for existence in our code Is that the random function generates only positive numbers ; and negative ones are important if a whole around spots are to be drawn around the center of operation

And the numeric explanation are:

1. Loop1 : is the first quarter (+xrandom, -yrandom)
2. Loop2 : is the second quarter (-xrandom, -yrandom)
3. Loop3 : is the third quarter (-xrandom, +yrandom)
4. Loop4 : is the fourth quarter (+xrandom, +yrandom)

The loops passes through the previous functions into doing its functions and finally after the 4 loops are done the code returns to the point where the status of the program is checking the status of the keyboard and mouse and whether to keep using the bakhakh nor not.

5-What did we achieved and what are you did not achieve

Praise be to God we have achieved all the desired tasks, and we managed to add some desired tasks such as rectangle, quad , triangle, and the sweet SPRAY .