به نام خدا

گرافیک در محیط VC++

نگارش 1.0.2

مجتبی ابراهیمی

ebrahimi_shahed@yahoo.com

اگر اشکالاتی در جزوه می بینید آنها را میل کنید تا مورد بررسی و تجدید نظر قرار گیرند.

فهرست

مقدمه

در C++ استاندارد هیچ حالت گرافیکی برای نمایش اطلاعات در نظر گرفته نشده است. ولی توابع کتابخانه ای وجود دارند که با اضافه کردن آنها می توان برنامه های گرافیکی نوشت.در بعضی کامپایلرها این کتابخانه ها به صورت پیش فرض اضافه شده اند مثلا در TC++ کتابخانه graphics.h دارای توابع گرافیکی می باشد. محیط VC++ در حالت پیش فرض هیچ کتابخانه ای برای کارهای گرافیکی ندارد و شما باید کتابخانه winbgim.h را به برنامه هایتان آن اضافه کنید.

در حالت کلی در برنامه های console ما برای نمایش اطلاعات می توانیم از دو حالت استفاده کنیم :
الف) حالت متنی که تا به حال خروجی همه برنامه هایمان بدین صورت بود.
ب) حالت گرافیکی
که هر کدام از این حالت ها دارای پنجره مخصوص خودشان می باشند و شما می توانید در یک برنامه به طور همزمان با هر دو حالت نمایش کار کنید یعنی در حالت گرافیکی شکل هایتان را بکشید و در حالت متنی از کاربر داده ها را بگیرید یا چاپ کنید.

پیکسل : کوچکترین واحد قابل دسترسی در حالت گرافیکی می باشد.

شما برای اینکه بتوانید وارد حالت گرافیکی شود باید بتوانید یک پنجره ایجاد کنید . هر پنجره با استفاده از ارتفاع و پهنای آن مشخص می شود یعنی ما در راستای محور عمودی و افقی حداکثر می توانیم چند پیکسل را آدرس دهی کنیم.مثلا پنجره ای با ارتفاع 300 یعنی پنجره ای که در راستای عمودی 300 پیکسل دارد که از بالا به پایین از صفر تا 299 شماره گذاری می شوند.

شفافیت: به حاصل ضرب پهنا در ارتفاع شفافیت می گویند.مثلا 800×600 یعنی پنجره ما دارای ارتفاع 800 پیکسل و پهنای 600 پیکسل است.

در حالت گرافیکی نیز مانند حالت متنی یک مکان نما داریم ولی در این حالت مکان نما دیده نمی شود.

دستورات اولیه

initwindow

این دستور برای ایجاد یک پنجره و ورود به حالت گرافیکی استفاده می شود و شکل کلی آن به صورت زیر است.

void initwindow(int width , int height)

برای مثال دستور initwindow(400 ,500) یک صفحه گرافیکی با عرض 400 و ارتفاع 500 ایجاد می کند.

closegraph

هر وقت که ما یک پنجره گرافیکی ایجاد می کنیم سیستم برای آن حافظه در نظر می گیرد . تابع closegraph پس از بستن پنجره حافظه گرفته شده را آزاد می کند.

void closegraph()

delay

این تابع یک عدد صحیح به عنوان آرگومان می گیرد وقتی برنامه به این دستور می رسد برنامه با توجه به عدد برنامه را برحسب میلی ثانیه متوقف می کند.

void delay(unsigned int n)

برای مثال دستور delay(50) موجب می شود برنامه به اندازه پنجاه میلی ثانیه متوقف شود.

مثال : برنامه ای که محیط گرافیکی را باز کرده و پس از 3 ثانیه می بندد.

```
// graphic start.cpp

#include "stdafx.h"
#include "winbgim.h"

int main(){
        initwindow( 400 , 400 );
        delay( 3000 );
        closegraph();
        return 0;
}
```

تابع های برای کشیدن اشکال هندسی

line

این تابع مختصات دو نقطه را به عنوان آرگومان می گیرد و خطی بین دو نقطه رسم می کند.شکل کلی این تابع به صورت زیر است :

```
void line( int x0 , int y0 . int x1, int y1)
```

که خطی از نقطه (x0,y0) به نقطه (x1,y1) می کشد.

مثال : برنامه زیر با استفاده از تابع line یک مثلث می کشد.

```
// line.cpp

#include "stdafx.h"
#include "winbgim.h"
#include <conio.h>

int main(){
	initwindow( 400 , 400 );
	line ( 100 , 100 , 200 , 200 );
	line ( 100 , 100 , 200 , 100 );
	line ( 200 , 200 , 200 , 100 );
	getch();
	closegraph();
	return 0;
}
```

اگر دقت کرده باشید این برنامه ها خط ها را با رنگ سفید می کشد ، رنگ سفید رنگ پیش فرض برای کشیدن اشکال است.برای تغییر رنگ از دستور setcolor استفاده می کنیم.

setcolor

این تابع باعث می شود که رنگ شکل هایی که بعد از این دستور میکشیم به رنگی در آید که به عنوان آرگومان به این تابع می فرستیم.

در حالت استاندارد ما 16 رنگ داریم که متناظر با هر رنگ یک شماره داریم که برای اعلان رنگ از نام رنگ یا شماره آن استفاده می کنیم.

0	BLACK
1	BLUE
2	GREEN
3	CYAN
4	RED
5	MSGENTA
6	BROWN
7	LIGHTGRAY
8	DARKGRAY
9	LIGHTBLUE
10	LIGHTGREEN
11	LIGHTCYAN
12	LIGHTRED
13	LIGHTMSGENTA
14	YELLOW
15	WHITE

شکل کلی این دستور به صورت زیر می باشد

void setcolor(int color)

که color شماره رنگ یا نام رنگ دقیقا مثل جدول بالا می باشد.برای مثال دستور setcolor(2) باعث می شود شکل هایی که بعد از این دستور می کشیم سبز باشند مگر اینکه با دستور setcolor دوباره رنگ را عوض کنیم.

حال اگر بخواهید یک رنگ دلخواه را به عنوان رنگ خود انتخاب کنید باید از دستور زیر استفاده کنید.

COLOR(RGB)

این دستور برای انتخاب رنگ دلخواه استفاده می شود ، در این حالت شما مقدار رنگ های قرمز و آبی و سبز را از 0 تا 255 مشخص می کنید و این تابع رنگ حاصل از ترکیب این سه رنگ را برمی گرداند که شما می توانید در تابع setcolor یا دیگر تابع هایی که با رنگ کار می کنند از آن استفاده کنید.

long int COLOR(int RED , int GREEN , int BLUE)

مثال: این برنامه در خروجی خود یک طیف رنگ پیوسته را به نمایش می گذارد.

// COLOR.cpp

```
#include "stdafx.h"
#include "winbgim.h"

void main(){
        initwindow(400,400);
        for(int i=0 ; i<255 ; i++ ){
                setcolor(COLOR(i,150,256-i));
                line(100, 100+i ,200 , 100+i );
                delay(30);
        }
        getch();
   closegraph( );
}
```

putpixel

یک pixel را با رنگ دلخواهی رنگ می کند.

```
void putpixel ( int  x , int y , int color)
```

که نقطه (x,y) مختصات نقطه و color رنگ مورد نظر می باشد که با اعداد 0 تا 16 یا دستور COLOR مشخص می شود.
مثلا دستور putpixel(10 , 20 ,1) نقطه (10 , 20) را با آبی رنگ می کند.

مثال: برنامه ای که با استفاده از دستور putpixel خطی را از (100,100) به (200,200) می کشد.

```
#include "stdafx.h"
#include "winbgim.h"
#include <conio.h>

void main(){
        initwindow(400,4000;
        for(int i=100 ; i<=200 ; i++ )
                putpixel( i , i , RED );
getch();
closegraph();
}
```

moveto

این تابع موجب انتقال مکان نما به یک نقطه دلخواه می شود.
void moveto (int x , int y)
که مکان نما را به نقطه (x,y) می برد.

moverel

جای مکان نما را به صورت نسبی تغییر می دهد.
void moverel (int dx , int dy)

اگر مکان اولیه مکان نما (x0 , y0) باشد آنرا به نقطه (x0+dx , y0+dy) منتقل می کند.
برای مثال دستور های مقابل موجب انتقال مکان نما به نقطه (90 , 110) می شود.
moveto(100 , 100);
moverel(10 , -10);

lineto , linerel

void lineto(int x , int y)
تابع lineto خطی را از جایی که مکان نما در آن قرار دارد به (x,y) می کشد و مکان نما را به نقطه (x,y) می برد.

void linerel(int dx , int dy)
اگر مکان نما در (x0 , y0) باشد از محل مکان نما خطی به نقطه (x0+dx , y0+dy) می کشد و مکان نما را نیز به نقطه دوم می برد.

مثال : رسم مثلث با استفاده از lineto

```
#include "stdafx.h"
#include "winbgim.h"
#include <conio.h>

void main(){
        initwindow(400 , 400);
        setcolor ( COLOR (125 , 12 , 200));
        moveto( 100 , 100 );
        lineto( 200 , 200 );
        lineto( 100 , 200 );
        lineto( 100 , 100);
        getch();
        closegraph();
        }
```

تابع line برعکس توابع lineto و linerel جای مکان نما را تغییر نمی دهد.

rectangle

از این تابع برای رسم مستطیل تو خالی استفاده می شود.
void rectangle(int left , int top , int right , int bottom)
که آرگومان های آن به ترتیب سمت چپ و بالای مستطیل و سمت راست و پایین مستطیل می باشد.

مثال : برنامه زیر با استفاده از تابع rectangle یک کادر می کشد.

```
// Kadr.cpp

#include "stdafx.h"
#include "winbgim.h"
#include <conio.h>

int main(){
        initwindow( 500 , 500 );
        setcolor(14);
        for ( int i=0 ; i<50 ; i++ ){
                rectangle( 100+i , 100+2*i , 400-i , 400-2*i);
                delay(60);
        }
        getch();
        closegraph();
        return 0;
}
```

circle

برای رسم دایره تو خالی استفاده می شود :
void circle (int x , int y , int radius)

که دایره ای به مرکز (x,y) و شعاع radius رسم می کند.

مثال : زسم دایره توپر با استفاده از تابع circle

```
// fill color.cpp
```

```
#include "stdafx.h"
#include "winbgim.h"
#include <conio.h>

int main(){
	initwindow( 510 , 510 );
	for( int i=0 ; i<255 ; i++ ){
		setcolor(COLOR( 256-i , i , 256-i ));
		circle(255,255,i);
	}
	getch();
	closegraph();
	return 0;
}
```

البته اگر دقت کرده باشید باز هم نقطه هایی داخل دایره هستند که رنگ نشده اند.

cleardevice

این دستور کلیه شکلهای روی صفحه را پاک می کند.

```
void cleardevice()
```

مثال : دایره ای که در حال بزرگ شدن است!!

```
// rising circle.cpp

#include "stdafx.h"
#include "winbgim.h"

int main()
{
	initwindow( 500 , 500 );
	for(int i=0 ; i<250 ; i++ ){
		circle(250 , 250 , i );
		circle(250 , 250 , i+1 );
		circle(250 , 250 , i+2 );
		delay(40);
		setcolor( COLOR ( 144 , 144 , i ) );
		cleardevice();
	}
	closegraph();
	return 0;
```

}

bar

این تابع برای رسم مستطیل توپر استفاده می شود و آرگومان هایش مثل تابع rectangle است.

void bar(int left , int top , int right , int down)

رنگ شکل هایی که ذاتاً توپر هستند مثل bar و bar3d و ... با تابع setfillstyle مشخص می شود .

setfillstyle

این تابع برای رنگ کردن شکلهای توپر مثل bar و ellipse استفاده می شود و شکل کلی آن به صورت زیر است.

void setfillstyle(int pattern , int color)

که آرگومان دوم آن مثل آرگومان تابع setcolor میتواند شماره بگیرد یا از تابع COLOR استفاده کند. آرگومان اول آن حالت پر کردن شکل است که طبق جدول زیر مشخص می شود.

EMPTY_FILL	0
SOLID_FILL	1
LINE_FILL	2
LTSLASH_FILL	3
SLASH_FILL	4
BKSLASH_FILL	5
LTBKSLASH_FILL	6
HATCH_FILL	7
XHATCH_FILL	8
INTERLEAVE_FILL	9
WIDE_DOT_FILL	10
CLOSE_DOT_FILL	11
USER_FILL	12

مثال : برنامه ای که یک مستطیل را با تمام حالت های پر کردن بکشد.

```
// fillpatern.cpp

#include "stdafx.h"
#include "winbgim.h"

int main(){
	initwindow( 400 , 400 );
	for( int i=0 ; i<12 ; i++ ){
		setfillstyle( i , 2 );
		bar( 100 , 100 , 300 , 300 );
		delay(1500);
	}
	closegraph();
	return 0;
}
```

bar3d

این تابع برای رسم مکعب است .

void bar3d(int left , int top , int right , int down , int depth , int topflag)

که چهار آرگومان اول برای کشیدن وجهی است که به طرف صفحه نمایش است و depth برای عمق شکل است. آرگومان آخر اگر یک باشد دو خط بالا نشان داده می شوند ولی اگر صفر باشد نشان داده نمی شوند.

رنگ قسمت توپر مکعب با تابع setfillstyle و رنگ خط های کناره با تابع setcolor نمایش داده می شود.

مثال : برنامه که حالت های مختلف آرگومان های مکعب را نشان می دهد.

```
// bar3d example.cpp

#include "stdafx.h"
#include "winbgim.h"
#include <conio.h>

int main(){
	initwindow(500 , 500);
	setfillstyle( 9 , 2 );
```

```
        bar3d( 100 , 100 , 200 , 200 , 50 , 1 );
        bar3d( 300 , 100 , 400 , 200 , 50 , 0 );
        bar3d( 100 , 300 , 200 , 400 , -50 , 1 );
        bar3d( 300 , 300 , 400 , 400 , -50 , 0 );
        getch();
        closegraph();
        return 0;
}
```

getx , gety , getmaxx , getmaxy , getmaxcolor

شکل کلی این توابع به صورت زیر است :

```
int getx();
int gety();
int getmaxx();
int getmaxy();
int getmaxcolor();
```

تابع های getx و gety مختصات مکان نما را در مختصات گرافیکی بر می گردانند. تابع getmaxx پهنای صفحه و تابع getmaxy ارتفاع صفحه را برمی گرداند.تابع getmaxcolor تعداد ماکسیمم رنگ های (استاندارد) پشتیبانی شده را برمی گرداند.
البته تابع های getmaxx و getmaxy بعضی وقت ها مقدار کاملا دقیق را برنمی گردانند.

مثال: این تابع توابع بالا را فراخوانی می کند و مقدار بازگشتی آنها را در صفحه نمایش چاپ می کند.

```
// getmax.cpp

#include "stdafx.h"
#include "winbgim.h"
#include <conio.h>

int main(){
        initwindow(200 , 500);
        moveto(100 , 10);
        printf("x=%d , y=%d\n" , getx() , gety() );
        printf("maxx=%d  , maxy=%d \n" , getmaxx() , getmaxy() );
        printf("maxcolor=%d",getmaxcolor() );
        closegraph();
        getch();
        return 0;
}
```

مثال : برنامه ای که اندازه یک مستطیل را با تغییرات اندازه پنجره تغییر می دهد.

```
// rectangli fit.cpp

#include "stdafx.h"
#include "winbgim.h"

int main(){
initwindow(400 , 400);
do{
        cleardevice();
        rectangle( getmaxx()/10 , getmaxy()/10 , 9*getmaxx()/10 , 9*getmaxy()/10 );
        delay(20);
        if ( kbhit() )
                break;
}while(1);
closegraph();
return 0;
}
```

اگر شما پس از اجرای برنامه پنجره خروجی را بزرگ یا کوچک کنید مستطیل نیز بزرگ و کوچک خواهد شد.

مثال : برنامه حرکت یک مستطیل توپر در صفحه با استفاده از کلید های جهت دار طوری که مستطیل از صفحه خارج نشود.

```
//    moving bar2.cpp
//    Mojtaba Ebrahimi
//    Ebrahimi_shahed@yahoo.com

//    Using arrow keys to move the bar ;
//    Using R key to reset bar ;
//    The bar go not out of window!!

#include "stdafx.h"
#include "winbgim.h"


int main()
{
        initwindow(800,600);
        setfillstyle( 7 , 2 );

        char ch;
        int xposition=getmaxx()/2-100;
```

```
        int yposition=getmaxy();
        do {
                ch=0;
                if (  kbhit() )
                        ch=getch();
                setfillstyle( 7 , 0 );
                bar ( xposition , yposition/2-30 , xposition+200, yposition/2+30 );
                switch ( ch ) {
                case 72:
                        if( yposition/2-30 > 0  )
                                yposition-=10;
                                break;
                case 75:
                        if (xposition > 0 )
                        xposition-=10;
                        break;
                case 77:
                        if (xposition+200 < getmaxx() )
                                xposition+=10;
                                break;
                case 80:
                        if( yposition/2+30 < getmaxy())
                                yposition+=10;
                                break;
                case 'r':
                case 'R':
                        xposition=getmaxx()/2-100;
                        yposition=getmaxy();
                        break;
                }
                setfillstyle( 7 , 2 );
                bar ( xposition , yposition/2-30 , xposition+200, yposition/2+30 );
                delay(10);
        }while ( ch != 27 );
        getch();
        return 0;
}
```

getcolor

این تابع آخرین رنگ تعیین شده توسط دستور setcolor را برمی گرداند و شکل کلی آن به صورت زیر است.

```
int getcolor();
```

getpixel

این تابع مختصات یک نقطه را از شما می گیرد و شماره رنگ آنرا برمی گرداند.

```
int getpixel( int x , int y);
```

مثال : تابعی که یک جعبه رنگ به شما نمایش می دهد و شما با دادن مختصات x و y در بازه 0 تا 255 رنگ مورد نظر خود را انتخاب می کنید و صفحه به آن رنگ در می آید.

```
// getpixel example.cpp

#include "stdafx.h"
#include "winbgim.h"
#include <conio.h>
#include <iostream.h>

int main(){

        initwindow(256 , 256);
        for( int i=0 ; i<256 ; i++ )
                for( int j=0 ; j<256 ; j++ ){
                        putpixel ( i , j , COLOR( i , i+j , j ) );
                }

        int x , y ;
        cout<<"Enter your color position:\nx=";
        cout.flush();
        cin>>x;
        cout<<"y=";
        cout.flush();
        cin>>y;
        setfillstyle(1 , getpixel ( x , y) );
        bar( 0 , 0 , getmaxx() , getmaxy() );
        getch();
        closegraph();

        return 0;
}
```

drawpoly, fillpoly

این توابع برای رسم چند ضلعی یا بهتر چند خطی هستند.شکل کلی این توابع یه صورت زیر است.

```
void drawpoly( int num_point , int  point_arr[] )
void fillpoly( int num_point , int  point_arr[] )
```

که num_point تعداد نقاط است و point_arr آرایه ای است که مختصات نقطه ها در آن قرار دارد که دو عنصر اول مختصات x و y نقطه اول و دو عنصر مختصات نقطه دوم و ... می باشد. آرایه حداقل باید دوبرابر num_point عنصر داشته باشد.
تابع drawpoly بین دو نقطه متوالی خطی می کشد .برای رسم چند ضلعی بسته باید مختصات نقطه آخر و اول یکی باشد .
تابع fillpoly نیز مانند تابع drawpoly عمل می کند ، با این تفاوت که داخل شکل را پر می کند.

مثال: برنامه ای که شکل های یک چند ضلعی را یکی یکی می کشد و در نهایت آنرا پر میکند.

```
// drawpoly.cpp

#include "stdafx.h"
#include "winbgim.h"
#include <conio.h>

int main(){
	initwindow( 400 , 400 );
	int a[]={ 100 ,100 , 150 ,200
		, 100 , 300 , 300 , 300
		, 250 , 200 , 300 , 100
		, 100 , 100 };
	setcolor(2);
	for( int i= 2 ; i<8 ; i++ ){
		drawpoly( i , a );
		getch();
	}
	setfillstyle( 9 , COLOR( 150 , 230 , 100) );
	fillpoly( 7 , a );
	getch();
	closegraph();
	return 0;
}
```

arc

این تابع برای کشیدن کمان استفاده می شود .
void arc(int x , int y , int start_angle , int end_angle , int radius)

که دو آرگومان اول مختصات مر کز دایره ای است که می خواهیم کمان آنرا بکشیم و دو آرگومان بعدی زاویه شروع و پایان کمان را مشخص می کند . آرگومان آخر هم شعاع دایره ای است که می خواهیم کمان آنرا بکشیم.

مثال : برنامه زیر یک دایره را به تدریج با استفاده از کمان رسم می کند.

```
// arc.cpp

#include "stdafx.h"
#include "winbgim.h"
#include <conio.h>

int main(){

	initwindow( 400 , 400 );
	setcolor( GREEN );
	for(int i=1 ; i<361 ; i++ ){
		arc(200 , 200 , 0 , i , 180 );
		delay(35);
	}
	getch();
	closegraph();
	return 0;
}
```

ellipse

این تابع برای رسم یک محدوده خاص از بیضی استفاده می شود. که رنگ این تابع با setcolor مشخص می شود.شکل کلی تابع به صورت زیر است:
void ellipse(int x, int y, int start_angle, int end_angle, int rx, int ry)

که x و y مرکز بیضی و rx شعاع در راستای محور x و ry شعاع در راستای محور y می باشد. start_angle و end_angle محدوده زاویه برحسب درجه است که می خواهیم رسم شود.

fillellipse

این تابع برای رسم بیضی تو پر استفاده می شود که رنگ آن با تابع setfillstyle مشخص می شود. شکل کلی این تابع به صورت زیر است :

void fillellipse(int x , int y , int xradiuse , int yradiuse)

که x و y مرکز بیضی و xradiuse شعاع x و yradiuse شعاع y می باشد.
برای رسم دایره توپر باید xradiuse و yradiuse را برابر قرار داد.

نمایش اطلاعات در حالت گرافیکی

شما در حالت گرافیکی فقط امکان چاپ رشته را دارید پس اگر بخواهید یک عدد یا مقدار یک متغیر را چاپ کنید باید آن را به رشته تبدیل کنید.

outtext
این تابع یک رشته را در محل مکان نما نمایش می دهد.
void outtext(char *string)

رنگ رشته با تابع setcolor مشخص می شود. تنظیمات font و اندازه متن توسط تابع settextstyle مشخص می شود.

settextstyle
این تابع برای تنظیمات متن های چاپ شده در محیط گرافیکی به کار می رود.
void settextstyle(int font , int direction , int size)

که آرگومان اول یکی از font های موجود در جدول زیر یا شماره آنها است:

0	DEFAULT_FONT
1	TRIPLEX_FONT
2	SMALL_FONT
3	SANSSERIF_FONT
4	GOTHIC_FONT

البته در VC++ تعداد فونت ها 12 تاست ولی فونت های بالا بین تمام کامپایلر های گرافیکی BGI مشترک هستند.

و آرگومان دوم جهت تعیین جهت چاپ متن است که صفر برای افقی و یک برای عمودی است.

آرگومان سوم اندازه متن است که یکی از اعداد یک تا ده می باشد.

مثال: برنامه که متنی را با تمام font های موجود نمایش می دهد.

```
// font.cpp

#include "stdafx.h"
#include "winbgim.h"
#include <conio.h>


int main(){
	initwindow(400 , 600 );
	for( int i=0; i<12 ; i++ ){
		setcolor(i+1);
		settextstyle( i , 0 , 4);
		moveto(0 , 50*i);
		outtext("Example text");
	}
	getch();
	closegraph();
	return 0;
}
```

outtextxy

این تابع برای درج یک رشته در یک مکان خاص به کار می رود.

```
void outtextxy(int x , int y , char *string)
```

که دو آرگومان اول مختصات نقطه ای است که می خواهیم متن را نمایش دهیم و آرگومان سوم رشته ای است که می خواهیم نمایش دهیم.

مثال : برنامه که رنگ متنی را به دو شکل تغییر می دهد که در حالت دوم به صورت چشمک زن در می آید.

```
// outtextxy.cpp

#include "stdafx.h"
```

```
#include "winbgim.h"

int main(){

	initwindow( 500 , 400 );
	int Red=100 , Green=20 , Blue=48 ;
	settextstyle( 4 , 0 , 4 );

	do{
		setcolor( COLOR ( Red+=7 , Green-=11 , Blue-=11 ) );
		outtextxy( 50  , 200 , "C++ Programming");
		delay(20);
	}while( !kbhit() );

	getch();

	// blinking text

	int counter=0;
	do{
		if ( counter%2 == 0)
			setcolor( 14 );
		else
			setcolor( 0 );
		counter++;
		delay(100);
		outtextxy( 50 , 200 , "C++ Programming");
	}while( !kbhit() );

	closegraph();
	return 0;
}
```

کار با mouse

حالت های mouse

همانطور که برای رنگ و حالت پر کردن آرگومان های از پیش تعیین شده ای داشتیم برای ماوس نیز از این آرگومان ها استفاده می کنیم.

WM_MOUSEMOVE	حرکت ماوس
WM_LBUTTONDBLCLK	دابل کلیک با دکمه چپ
WM_LBUTTONDOWN	زدن دکمه چپ
WM_LBUTTONUP	رها کردن دکمه چپ
WM_MBUTTONDBLCLK	دابل کلیک با دکمه وسط
WM_MBUTTONDOWN	زدن دکمه وسط
WM_MBUTTONUP	رها کردن دکمه وسط
WM_RBUTTONDBLCLK	دابل کلیک با دکمه راست
WM_RBUTTONDOWN	زدن دکمه سمت راست
WM_RBUTTONUP	رها کردن دکمه سمت راست

registermousehandler

با استفاده از این تابع مشخص می کنیم که وقتی ما کاری را با ماوس انجام می دهیم مثلا راست کلیک می کنیم یا چپ کلیک می کنیم برنامه باید چه تابعی را اجرا کند. این تعریف باعث می شود که در طول اجرای برنامه هر وقت ما آمدیم عمل خاصی را انجام دادیم اجرای برنامه به تابعی که مشخص کردیم منتقل شود و بعد از اجرای تابع دوباره برنامه از جای قبلی به کار خود ادامه دهد.

void registermousehandler(int kind, void h(int x, int y))

آرگومان اول یکی از حالت های ماوس می باشد که از جدول حالت های ماوس باید انتخاب شود. و اگر در طول اجرای برنامه آن حالت اتفاق بیفتد تابع h که آرگومان دوم است فراخوانی می شود. آرگومان دوم نیز نام تابعی است که می خواهیم اجرا شود این تابع حتما باید دو آرگومان int داشته باشد.مختصات جایی که در آن کلیک (به هر صورت) کرده ایم به عنوان آرگومان ها به تابع فرستاده می شود.

مثال : برنامه ای که هر کجای صفحه چپ کلیک کنیم خط سفید و اگر دابل کلیک کنیم خط سبز و اگر کلیک راست کنیم خط قرمز از مرکز صفحه به آن نقطه بکشد. اگر دکمه وسط را بزنیم کلیه خطوط پاک شوند.

```
// mouse line.cpp

#include "stdafx.h"
#include "winbgim.h"
```

```
void whitelinedrawer(int x , int y){
	setcolor(15);
	line( getmaxx()/2 , getmaxy()/2 , x , y);
}

void redlinedrawer(int x , int y){
	setcolor(4);
	line( getmaxx()/2 , getmaxy()/2 , x , y);
}

void greenlinedrawer(int x , int y){
	setcolor(2);
	line( getmaxx()/2 , getmaxy()/2 , x , y);
}

void clearlines(int x , int y){
	cleardevice();
}

int main(){
	initwindow( 500 , 500 );
	registermousehandler( WM_LBUTTONDOWN , whitelinedrawer );
	registermousehandler( WM_LBUTTONDBLCLK , greenlinedrawer );
	registermousehandler( WM_RBUTTONDOWN , redlinedrawer   );
	registermousehandler( WM_MBUTTONDOWN , clearlines   );
	getch();
	closegraph();
	return 0;
}
```

mousex و mousey

این توابع مختصات موقعیت ماوس را در پنجره گرافیکی بر می گردانند که برای mousex مقداری بین 0 تا getmaxx() است.

```
int mousex();
int mousey();
```

ismouseclick

این تابع برای چک کردن اینکه حالت دلخواه ما اتفاق افتاده است یا نه استفاده می شود. شکل کلی این تابع به صورت زیر است:

bool ismouseclick(int kind);

آرگومان تابع یکی از انواع حالت های ماوس است اگر این حالت اتفاق افتاده باشد تابع TRUE و در غیر این صورت FALSE بر می گرداند.

getmouseclick

این تابع چک می کند آن نوع کلید خاصی که ما می خواهیم زده شده است یا نه.اگر زده شده باشد مختصات آنرا با دو آرگومان ارجاع بر می گرداند.در غیر این صورت در آرگومان های ارجاع مقدار 1- را قرار می دهد.

void getmouseclick(int kind, int& x, int& y)

که آرگومان اول حالت ماوس و آرگومان دوم وسوم ورودی های ارجاع برای بازگرداندن مختصات هستند.

مثال: این برنامه هر وقت شما کلید چپ ماوس را بزنید مختصات آنرا در خروجی متنی چاپ می کند و اگر دکمه ای از صفحه کلید را بزنید برنامه خاتمه پیدا می کند.

```
#include "winbgim.h"
#include <iostream.h>
#include <conio.h>

void main(){
        int x;
        int y;

        do{
                if ( ismouseclick( WM_LBUTTONDOWN )){
                getmouseclick( WM_LBUTTONDOWN , x , y );
                cout<<x
                        <<"\t"
                        <<y
                        <<endl;
                        }
                delay(10);
                }while ( !kbhit() );

   closegraph( );
}
```

clearmouseclick

این تابع مثل تابع getmouseclick عمل می کند ولی مقدار x و y را بر نمی گرداند.

مثال: این برنامه یک دایره قرمز را می کشد سپس شروع به رسم مثلث هایی می نماید اگر شما در ناحیه قرمز کلیک کنید برنامه خاتمه پیدا می کند.

```
#include "winbgim.h"
#include <iostream.h>

bool red_clicked = false;

void click_handler(int x, int y){
   if (getpixel(x,y) == RED)
      red_clicked = true;
         }

void triangle(int base, int height){
   int maxx = getmaxx( );
   int maxy = getmaxy( );
   line(maxx/2 - base/2, maxy - 10, maxx/2 + base/2, maxy - 10);
   line(maxx/2 - base/2, maxy - 10, maxx/2, maxy - 10 - height);
   line(maxx/2 + base/2, maxy - 10, maxx/2, maxy - 10 - height);
         }

void main(void){

   int maxx, maxy;  // Maximum x and y pixel coordinates
   int divisor;     // Divisor for the length of a triangle side
   // Put the machine into graphics mode and get the maximum coordinates:
   initwindow(800, 600);
   maxx = getmaxx( );
   maxy = getmaxy( );

   registermousehandler(WM_LBUTTONDOWN, click_handler);
   setfillstyle(SOLID_FILL, RED);
   setcolor(WHITE);
   fillellipse(maxx/2, maxy/2, 50, 50);

   // Print a message and wait for a red pixel to be double clicked:
   settextstyle(DEFAULT_FONT, HORIZ_DIR, 2);
   outtextxy(20, 20, "Left click in RED to end.");
   setcolor(BLUE);
   red_clicked = false;
   divisor = 2;
```

```
    while (!red_clicked){
        triangle(maxx/divisor, maxy/divisor);
                delay(500);
        divisor++;
                        }

    cout << "The mouse was clicked at: ";
    cout << "x=" << mousex( );
    cout << " y=" << mousey( ) << endl;
    closegraph( );
}
```

خرداد 1385