/\* Program PIXEL\_1.C

\*\*

\*\* P. H. Anderson, MSU, 27 Nov 94

\*/

#include <stdio.h>

#include <graphics.h>

void main(void)

{

int graphdriver = DETECT, graphmode, color, n, m;

initgraph(&graphdriver, &graphmode, "c:\\turboc");

/\* detect and initialize graphics system \*/

for(n=0; n<50; n++)

{

putpixel(250+n,350,BLUE); /\* draw a horizontal line \*/

}

for(m=0; m<5; m++)

{

for(n=0; n<5; n++)

{

putpixel(250+n,250+m,RED);

}

}

}

/\* Program LINE\_1.C

\*\*

\*\* Illustrates how to draw a line. Note that xy screen coords are

\*\* 0,0 (upper left) to 500,700 (lower right).

\*\*

\*\* Cheryl Melvin, 27 Sept, 30

\*\* P. H. Anderson, 30 Sept 93

\*\*

\*/

#include <stdio.h>

#include <graphics.h>

#include <dos.h>

void main(void)

{

int graphdriver = DETECT, graphmode, linestyle=0;

int i, x, y;

initgraph(&graphdriver, &graphmode, "c:\\turboc");

/\* detect and initialize graphics system

\*\* Note that when graphdriver is DETECT, the graphmode is auto

\*\* set to the highest resolution available.

\*\* "c:\\tc\\bgi" is the path full path name to the graphics

\*\* driver

setcolor(RED); /\* see program COLOR for full listing of colors\*/

outtextxy(10, 10, "Line Demo");

/\* outputs text at x, y using default font and size \*/

setcolor(GREEN);

/\* draw some horizontal lines \*/

for (i=0, y=0; i<10; i++)

{

setlinestyle(SOLID\_LINE, linestyle, NORM\_WIDTH);

/\* options are SOLID\_LINE, DOTTED\_LINE, CENTER\_LINE

\*\* and DASHED\_LINE. linestyle is for user defined line

\*\* styles.

\*\* widths are NORM\_WIDTH and THICK\_WIDTH

\*/

moveto(0,y);

lineto(700,y);

y += 50;

}

sleep(5); /\* pause to admire \*/

setcolor(MAGENTA);

for (i=0, x=0; i<10; i++)

{

setlinestyle(DASHED\_LINE, linestyle, THICK\_WIDTH);

/\* options are SOLID\_LINE, DOTTED\_LINE, CENTER\_LINE

\*\* and DASHED\_LINE. linestyle is for user defined line

\*\* styles.

\*\* widths are NORM\_WIDTH and THICK\_WIDTH

\*/

moveto(x,0);

lineto(x, 50+i\*50);

x += 70;

}

sleep(10);

closegraph();

}

/\* POLY.C

\*\*

\*\* Illustrates how to draw polygons. Note that vertices are definded

\*\* in an array in the form {x0, y0, x1, y1, ...}. Thus the array must

\*\* contain two times the number of points. For a closed polygon, the

\*\* last point must be the same as the first.

\*\*

\*\*

\*\* Cheryl Melvin, MSU, October 7, 1993

\*/

#include <stdio.h>

#include <graphics.h>

#include <conio.h>

main()

{

int driver, mode;

int shape[10] = {

10, 10,

100, 80,

200, 200,

350, 90,

10, 10

}; /\* closed polygon \*/

int shape1[6] = {

10, 10,

80, 60,

350, 90

}; /\* not closed \*/

driver = DETECT; /\* autotect \*/

mode = 0;

initgraph(&driver, &mode, "c:\\turboc");

setcolor(RED);

drawpoly(5, shape); /\* number of points, pointer to array \*/

setcolor(GREEN);

drawpoly(3, shape1);

getch(); /\* pause to admire \*/

restorecrtmode();

return 0;

}

/\* Program BAR\_1.C

\*\*

\*\* Illustrates how to draw a bar.

\*\*

\*\* Bar is drawn and colors are changed every second.

\*\*

\*\* P. H. Anderson, 25 Nov, 94

\*/

#include <stdio.h>

#include <graphics.h>

#include <dos.h>

void main(void)

{

int graphdriver = DETECT, graphmode, linestyle=0, color;

char string[80];

initgraph(&graphdriver, &graphmode, "c:\\turboc");

/\* detect and initialize graphics system \*/

setlinestyle(SOLID\_LINE, linestyle, NORM\_WIDTH);

for (color=0; color<16; color++)

{

setcolor(WHITE); /\* outline of bar is white \*/

setfillstyle(SOLID\_FILL, color);

/\* fill patterns are 0 - 11 \*/

bar3d(50, 50, 300, 200, 50, 1);

/\* bar3d(int startx, int starty, int endx, int endy,

\*\* int depth, int topflag. Note that if depth is 0

\*\* then bar is 2 dimensional, if topflag = 0, there is

\*\* no top

\*/

sprintf(string, "Color = %d", color);

setcolor(BLACK); /\* text is in black \*/

outtextxy(100, 100, string);

/\* outtextxy, (x, y, text \*/

sleep(1); /\* pause to admire \*/

}

closegraph();

}

/\* Program BAR\_2.C

\*\*

\*\* Illustrates bars with various fill patterns.

\*\*

\*\* Bar is drawn and colors are changed every second.

\*\*

\*\* P. H. Anderson, 30 Sept 93; Dec 9, '94

\*\*

\*/

#include <stdio.h>

#include <graphics.h>

#include <dos.h>

void main(void)

{

int graphdriver = DETECT, graphmode, linestyle=0, color;

int n;

char string[80];

initgraph(&graphdriver, &graphmode, "c:\\turboc");

/\* detect and initialize graphics system \*/

setlinestyle(SOLID\_LINE, linestyle, NORM\_WIDTH);

for (color=0; color<16; color++)

{

setcolor(RED); /\* outline of bar is red \*/

for (n=0; n<12; n++)

{

setfillstyle(n, color);

bar3d(50\*n+5, 50, 50\*n+50, 300, 0, 0);

sprintf(string, "%d", n);

outtextxy(50\*n+20, 310, string);

}

sprintf(string, "Color = %d", color);

setcolor(MAGENTA); /\* text is in magenta \*/

outtextxy(100, 400, string);

/\* outtextxy, (x, y, text \*/

sleep(1); /\* pause to admire \*/

}

closegraph();

}

/\* Program BAR\_3.C

\*\*

\*\* Illustrates how to draw a bar to represent a quantity.

\*\*

\*\* P. H. Anderson, 7 Dec 94

\*/

#include <stdio.h>

#include <graphics.h>

#include <dos.h>

/\* #define XXX \*/

void draw\_bar(int quan, int max\_quan, int fill\_color);

void main(void)

{

int graphdriver = DETECT, graphmode, linestyle=0, n;

char string[80];

initgraph(&graphdriver, &graphmode, "c:\\turboc");

/\* detect and initialize graphics system \*/

cleardevice();

setlinestyle(SOLID\_LINE, linestyle, NORM\_WIDTH);

for (n=0; n<200; n++) /\* make it go up \*/

{

draw\_bar(n, 200, GREEN);

delay(100);

}

for (n=200; n>=0; n--) /\* make it go down \*/

{

draw\_bar(n, 200, GREEN);

delay(100);

}

closegraph();

}

void draw\_bar(int quan, int max\_quan, int fill\_color)

{

int startx = 300, endx = 350, starty = 400, endy;

cleardevice();

/\* draw the new bar \*/

endy = (400 - (int) ((float) quan / max\_quan \* 400));

setcolor(WHITE);

bar3d(startx, starty, endx, endy, 0, 0);

setfillstyle(SOLID\_FILL, fill\_color);

floodfill (325, (starty + endy) / 2, WHITE);

#ifdef XXX

/\* erase any old bar \*/ /\* problem with flicker \*/

setcolor(BLACK);

bar3d(startx, endy, endx, 0, 0, 0);

setfillstyle(SOLID\_FILL, BLACK);

floodfill (325, endy-10, BLACK);

#endif

}

/\* Program BOX\_1.C

\*\*

\*\* Illustrates how to draw a box. Note that xy screen coords are

\*\* 0,0 (upper left) to 500,700 (lower right).

\*\*

\*\* P. H. Anderson, 30 Sept 93

\*\*

\*/

#include <stdio.h>

#include <graphics.h>

#include <dos.h>

void draw\_box (int startx, int starty, int endx, int endy, int color);

/\* draws an empty box with outline of lines having specified color \*/

void main(void)

{

int graphdriver = DETECT, graphmode, linestyle=0;

initgraph(&graphdriver, &graphmode, "c:\\turboc");

/\* detect and initialize graphics system \*/

setlinestyle(SOLID\_LINE, linestyle, THICK\_WIDTH);

draw\_box(50, 50, 300, 200, RED);

setcolor(GREEN);

/\* draw a cross in the box \*/

line(50, 50, 300, 200);

line(300, 50, 50, 200);

sleep(10); /\* pause to admire \*/

closegraph();

}

void draw\_box (int startx, int starty, int endx, int endy, int color)

{

setcolor(color);

rectangle(startx, starty, endx, endy);

}

/\*

\*\* Program CIRCLE\_1

\*\*

\*\* Illustrates how to draw circles.

\*\*

\*\* P. H. Anderson, MSU, 10 Nov, '94

\*/

#include <stdio.h>

#include <graphics.h>

#include <conio.h>

/\* this program draws a circle centered at x,y with a radius. \*/

void main(void)

{

int driver, mode, i;

driver = DETECT; /\* autotect\*/

mode = 0;

initgraph(&driver, &mode, "c:\\turboc");

setcolor(BLUE); /\*sets the color of the circle \*/

circle (350, 250, 200);

setcolor(RED);

circle (200, 200, 50);

setcolor(YELLOW);

circle (200, 200, 40);

setcolor(MAGENTA);

circle (200, 200, 30);

setcolor(GREEN);

circle (200, 200, 20);

getch(); /\* wait until key pressed \*/

restorecrtmode(); /\* restores the screen to the original screen \*/

}

/\*

\*\* Program CIRCLE\_2

\*\*

\*\* Draws an expanding circle with sound.

\*\* Note that the color of the circle is selected at random.

\*\*

\*\* Illustrates graphics, randomize, random, sound, nosound.

\*\*

\*\* P. H. Anderson, MSU, 10 Nov, '94

\*/

#include <stdio.h>

#include <stdlib.h>

#include <graphics.h>

#include <conio.h>

#include <time.h> /\* for randomize \*/

void main(void)

{

int driver, mode, n, r, color;

driver = DETECT; /\* autotect\*/

mode = 0;

initgraph(&driver, &mode, "c:\\turboc");

randomize();

/\* seeds random number generator \*/

while(1)

{

color = random(14)+1;

/\* note that random(n) returns a random number between

\*\* 0 and n-1. note that color is thus a random number in the

\*\* range 1-15, thus avoiding black

\*/

nosound();

for (n=0; n<50; n++)

{

setcolor(color); /\*sets the color of the circle \*/

r= (int) 200.0 \* n/50;

circle (350, 250, r);

sound((int)440.0/50\*n);

delay(20);

setcolor(BLACK);

circle (350, 250, r); /\*erase the circle\*/

}

}

}

/\* Program CIRCLE\_3.C

\*\*

\*\* Illustrates how to draw a line which turns in a circle

\*\*

\*\* Draws a circle in the center of the screen and a line moves

\*\* around the circle.

\*\*

\*\* P. H. Anderson, MSU, 10 Nov, '94

\*\*

\*/

#include <stdio.h>

#include <graphics.h>

#include <math.h>

#include <conio.h>

#define PI 3.14159

void draw\_polar\_line (int x\_1, int y\_1, int l, float theta);

/\* draws a line of length l at angle theta using the specified

\*\* line style and color from graphics coords x, y. angle theta is

\*\* in degrees.

\*/

float degrees\_to\_rads (float x);

void main(void)

{

int driver, mode, n;

float angle;

driver = DETECT; /\* autotect\*/

mode = 0;

clrscr();

initgraph(&driver, &mode, "c:\\turboc");

setcolor(BLUE); /\*sets the color of the circle \*/

circle (300, 250, 200); /\* x=300, y=250, r=200 \*/

setfillstyle (SOLID\_FILL, YELLOW);

floodfill (300, 250, BLUE);

/\* fills area in which x, y are located bounded by BLUE \*/

setlinestyle(SOLID\_LINE, 0, NORM\_WIDTH);

while(1)

{

for (n=0; n<90; n++)

{

angle = 360.0 \* n/90;

setcolor(RED);

draw\_polar\_line (300, 250, 200, angle);

delay(200);

setcolor(YELLOW); /\* erase the line \*/

draw\_polar\_line (300, 250, 200, angle);

}

}

restorecrtmode(); /\* restores the screen to the original screen \*/

}

void draw\_polar\_line (int x\_1, int y\_1, int l, float theta)

{

int diff\_x, diff\_y, x\_2, y\_2;

float angle\_rads;

angle\_rads = degrees\_to\_rads (theta);

diff\_x = (int) l \* cos (angle\_rads);

diff\_y = (int) l \* sin (angle\_rads);

x\_2 = x\_1 + diff\_x;

y\_2 = y\_1 - diff\_y; /\* note that up is minus \*/

line (x\_1, y\_1, x\_2, y\_2);

}

float degrees\_to\_rads (float x)

{

return(PI/180.0\*x);

}

/\* Program CIRCLE\_4

\*\*

\*\* Illustrates function floodfill

\*\*

\*\* Draws two concentric circles. Inner circle filled with GREEN.

\*\* Outer area filled with YELLOW. Various fill styles shown in inner

\*\* circle.

\*\*

\*\* P. H. Anderson, MSU, 10 Nov, '94

\*/

#include <stdio.h>

#include <graphics.h>

#include <math.h>

#include <conio.h>

void main(void)

{

int driver, mode, n;

float angle;

char string[80];

driver = DETECT; /\* autotect\*/

mode = 0;

clrscr();

initgraph(&driver, &mode, "c:\\turboc");

setcolor(BLUE); /\*sets the color of the circle \*/

circle (300, 250, 150); /\* x=300, y=250, r=150 \*/

circle (300, 250, 200);

/\* thus, there are now two closed areas bounded by blue \*/

setfillstyle (SOLID\_FILL, YELLOW);

floodfill (300, 75, BLUE);

/\* fills area in which x, y are located bounded by BLUE \*/

/\* note that this is the outer circle \*/

while(1)

{

for (n=0; n<=12; n++)

{

setfillstyle(n, GREEN);

/\* illustrates various fill styles \*/

floodfill(300, 250, BLUE); /\* inner circle \*/

setcolor(MAGENTA);

sprintf(string, "Fill Style: %d", n);

outtextxy(250, 25, string);

sleep(1);

setcolor(BLACK); /\* erase the text \*/

sprintf(string, "Fill Style: %d", n);

outtextxy(250, 25, string);

}

}

restorecrtmode(); /\* restores the screen to the original screen \*/

}

/\* Program CIRCLE\_5.C

\*\* Draws up to ten circles at predefined points.

\*\* Illustrates use of structures.

\*/

#include <stdio.h>

#include <graphics.h>

#include <conio.h>

struct Circle

{

int x;

int y;

int r;

}; /\* note global definition of the structure \*/

void draw\_n\_circles(struct Circle \*cir, int num);

void main(void)

{

int driver, mode, n;

struct Circle circles[10];

/\* initialize the possible circles \*/

for (n=0; n<10; n++)

{

circles[n].y = 250; /\* midway down the screen \*/

circles[n].r = 20;

circles[n].x = 70 + 700\*n/10;

}

driver = DETECT; /\* autotect\*/

mode = 0;

initgraph(&driver, &mode, "c:\\turboc");

setcolor(BLUE); /\*sets drawing color \*/

draw\_n\_circles(circles, 4); /\* draw 4 circles \*/

getch(); /\* wait until key pressed \*/

cleardevice(); /\* get rid of old image \*/

draw\_n\_circles(circles, 8); /\* draw 8 circles \*/

getch(); /\* wait until key pressed \*/

restorecrtmode(); /\* restores the screen to the original screen \*/

}

void draw\_n\_circles(struct Circle \*cir, int num)

{

int n;

setfillstyle (SOLID\_FILL, YELLOW);

for(n=0; n<num; n++)

{

circle(cir[n].x, cir[n].y, cir[n].r);

floodfill (cir[n].x, cir[n].y, BLUE);

}

}

1. Nacrtaj osnovne geometriske likove.

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT,gm,left=100,top=100,right=200,bottom=200,x= 300,y=150,radius=50;

initgraph(&gd, &gm, "C:\\TC\\BGI");

rectangle(left, top, right, bottom);

circle(x, y, radius);

bar(left + 300, top, right + 300, bottom);

line(left - 10, top + 150, left + 410, top + 150);

ellipse(x, y + 200, 0, 360, 100, 50);

outtextxy(left + 100, top + 325, "My first C graphics program");

getch();

closegraph();

return 0;

}

1. Auto u pokretu

#include <graphics.h>

#include <dos.h>

#include <conio.h>

main()

{

int i, j = 0, gd = DETECT, gm;

initgraph(&gd,&gm,"C:\\TC\\BGI");

settextstyle(DEFAULT\_FONT,HORIZ\_DIR,2);

outtextxy(25,240,"Press any key to view the moving car");

getch();

setviewport(0,0,639,440,1);

for (i = 0; i <= 420; i = i + 10, j++)

{

rectangle(50+i,275,150+i,400);

rectangle(150+i,350,200+i,400);

circle(75+i,410,10);

circle(175+i,410,10);

setcolor(j);

delay(100);

if (i == 420)

break;

clearviewport();

}

getch();

closegraph();

return 0;

}

1. Smješko

#include<graphics.h>

#include<conio.h>

#include<stdlib.h>

main()

{

int gd = DETECT, gm, area, temp1, temp2, left = 25, top = 75;

void \*p;

initgraph(&gd, &gm, "C:\\TC\\BGI");

setcolor(YELLOW);

circle(50, 100, 25);

setfillstyle(SOLID\_FILL, YELLOW);

floodfill(50, 100, YELLOW);

setcolor(BLACK);

setfillstyle(SOLID\_FILL, BLACK);

fillellipse(44, 85, 2, 6);

fillellipse(56, 85, 2, 6);

ellipse(50, 100, 205, 335, 20, 9);

ellipse(50, 100, 205, 335, 20, 10);

ellipse(50, 100, 205, 335, 20, 11);

area = imagesize(left, top, left + 50, top + 50);

p = malloc(area);

setcolor(WHITE);

settextstyle(SANS\_SERIF\_FONT, HORIZ\_DIR, 2);

outtextxy(155, 451, "Smiling Face Animation");

setcolor(BLUE);

rectangle(0, 0, 639, 449);

while(!kbhit())

{

temp1 = 1 + random (588);

temp2 = 1 + random (380);

getimage(left, top, left + 50, top + 50, p);

putimage(left, top, p, XOR\_PUT);

putimage(temp1 , temp2, p, XOR\_PUT);

delay(100);

left = temp1;

top = temp2;

}

getch();

closegraph();

return 0;

}

1. Pravokutnik

#include <graphics.h>

#include <conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:\\TC\\BGI");

setcolor(YELLOW);

rectangle(0,30,639,450);

settextstyle(SANS\_SERIF\_FONT,HORIZ\_DIR,2);

setcolor(WHITE);

outtextxy(275,0,"Bar Chart");

setlinestyle(SOLID\_LINE,0,2);

line(100,420,100,60);

line(100,420,600,420);

line(90,70,100,60);

line(110,70,100,60);

line(590,410,600,420);

line(590,430,600,420);

outtextxy(95,35,"Y");

outtextxy(610,405,"X");

outtextxy(85,415,"O");

setfillstyle(LINE\_FILL,BLUE);

bar(150,100,200,419);

setfillstyle(XHATCH\_FILL,RED);

bar(225,150,275,419);

setfillstyle(WIDE\_DOT\_FILL,GREEN);

bar(300,200,350,419);

setfillstyle(INTERLEAVE\_FILL,MAGENTA);

bar(375,125,425,419);

setfillstyle(HATCH\_FILL,BROWN);

bar(450,175,500,419);

getch();

return 0;

}

1. Bojanje dijelova

#include<graphics.h>

#include<conio.h>

int main()

{

int gd = DETECT, gm, midx, midy;

initgraph(&gd, &gm, "C:\\TC\\BGI");

setcolor(MAGENTA);

rectangle(0,40,639,450);

settextstyle(SANS\_SERIF\_FONT,HORIZ\_DIR,2);

setcolor(WHITE);

outtextxy(275,10,"Pie Chart");

midx = getmaxx()/2;

midy = getmaxy()/2;

setfillstyle(LINE\_FILL,BLUE);

pieslice(midx, midy, 0, 75, 100);

outtextxy(midx+100, midy - 75, "20.83%");

setfillstyle(XHATCH\_FILL,RED);

pieslice(midx, midy, 75, 225, 100);

outtextxy(midx-175, midy - 75, "41.67%");

setfillstyle(WIDE\_DOT\_FILL,GREEN);

pieslice(midx, midy, 225, 360, 100);

outtextxy(midx+75, midy + 75, "37.50%");

getch();

return 0;

}

1. 3d bar

#include <graphics.h>

#include <conio.h>

int main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:\\TC\\BGI");

setcolor(YELLOW);

rectangle(0,30,639,450);

settextstyle(SANS\_SERIF\_FONT,HORIZ\_DIR,2);

setcolor(WHITE);

outtextxy(275,0,"Bar Chart");

setlinestyle(SOLID\_LINE,0,2);

line(100,420,100,60);

line(100,420,600,420);

line(90,70,100,60);

line(110,70,100,60);

line(590,410,600,420);

line(590,430,600,420);

outtextxy(95,35,"Y");

outtextxy(610,405,"X");

outtextxy(85,415,"O");

setfillstyle(LINE\_FILL,BLUE);

bar(150,100,200,419);

setfillstyle(XHATCH\_FILL,RED);

bar(225,150,275,419);

setfillstyle(WIDE\_DOT\_FILL,GREEN);

bar(300,200,350,419);

setfillstyle(INTERLEAVE\_FILL,MAGENTA);

bar(375,125,425,419);

setfillstyle(HATCH\_FILL,BROWN);

bar(450,175,500,419);

getch();

return 0;

}

1. Crtanje krugova u bojama

#include<graphics.h>

#include<conio.h>

#include<dos.h>

main()

{

int gd = DETECT, gm, x, y, color, angle = 0;

struct arccoordstype a, b;

initgraph(&gd, &gm, "C:\\TC\\BGI");

delay(2000);

while(angle<=360)

{

setcolor(BLACK);

arc(getmaxx()/2,getmaxy()/2,angle,angle+2,100);

setcolor(RED);

getarccoords(&a);

circle(a.xstart,a.ystart,25);

setcolor(BLACK);

arc(getmaxx()/2,getmaxy()/2,angle,angle+2,150);

getarccoords(&a);

setcolor(GREEN);

circle(a.xstart,a.ystart,25);

angle = angle+5;

delay(50);

}

getch();

closegraph();

return 0;

}

1. Semafor

#include<graphics.h>

#include<conio.h>

#include<dos.h>

#include<stdlib.h>

main()

{

int gd = DETECT, gm, midx, midy;

initgraph(&gd, &gm, "C:\\TC\\BGI");

midx = getmaxx()/2;

midy = getmaxy()/2;

setcolor(RED);

settextstyle(SCRIPT\_FONT, HORIZ\_DIR, 3);

settextjustify(CENTER\_TEXT, CENTER\_TEXT);

outtextxy(midx, midy-10, "Traffic Light Simulation");

outtextxy(midx, midy+10, "Press any key to start");

getch();

cleardevice();

setcolor(WHITE);

settextstyle(DEFAULT\_FONT, HORIZ\_DIR, 1);

rectangle(midx-30,midy-80,midx+30,midy+80);

circle(midx, midy-50, 22);

setfillstyle(SOLID\_FILL,RED);

floodfill(midx, midy-50,WHITE);

setcolor(BLUE);

outtextxy(midx,midy-50,"STOP");

delay(2000);

graphdefaults();

cleardevice();

setcolor(WHITE);

rectangle(midx-30,midy-80,midx+30,midy+80);

circle(midx, midy, 20);

setfillstyle(SOLID\_FILL,YELLOW);

floodfill(midx, midy,WHITE);

setcolor(BLUE);

outtextxy(midx-18,midy-3,"READY");

delay(2000);

cleardevice();

setcolor(WHITE);

rectangle(midx-30,midy-80,midx+30,midy+80);

circle(midx, midy+50, 22);

setfillstyle(SOLID\_FILL,GREEN);

floodfill(midx, midy+50,WHITE);

setcolor(BLUE);

outtextxy(midx-7,midy+48,"GO");

setcolor(RED);

settextstyle(SCRIPT\_FONT, HORIZ\_DIR, 4);

outtextxy(midx-150, midy+100, "Press any key to exit...");

getch();

closegraph();

return 0;

}