גירסה 1.00 – 16.4.2004

גרפיקה ב-Borland C - יצירת שעון

במסמך זה נכיר חלק מהפונקציות הגרפיות של Borland C, ונראה איך נבנה בעזרתן שעון מחוגים קטן.

Borland C תומכת בשני מצבי עבודה: מצב טקסט ומצב גרפי. במצב הטקסט אנחנו משתמשים בפונקציות כגון printf כדי להדפיס תווים שונים על המסך.
במצב הגרפי אנחנו יכולים להציג גרפיקות רחבות יותר, כגון קווים, עיגולים וצורות נוספות. כמו כן נוכל גם להדפיס תווים על המסך, אם כי לא בעזרת printf.
נציג במסמך זה פונקציות גרפיות שונות ואת השימוש בהן.

מסמך זה הורד מהאתר http://underwar.livedns.co.il.
אין להפיץ מסמך זה במדיה כלשהי, ללא אישור מפורש מאת המחבר.
מחבר המסמך איננו אחראי לכל נזק, ישיר או עקיף, שיגרם עקב השימוש במידע המופיע במסמך, וכן לנכונות התוכן של הנושאים המופיעים במסמך. עם זאת, המחבר עשה את מירב המאמצים כדי לספק את המידע המדויק והמלא ביותר.

כל הזכויות שמורות ל**ניר אדר**

Nir Adar
Email: underwar@hotmail.com
Home Page: http://underwar.livedns.co.il

אנא שלחו תיקונים והערות אל המחבר.

גרפיקה ב-Borland C - יצירת שעון

פונקציות גרפיות

כדי להשתמש בפונקציות גרפיות, נכלול את הספריה graphics.h.

נתאר את הפונקציות הגרפיות בהן נשתמש:

כניסה למצב גרפי – initgraph()

הפונקציה initgraph מעבירה אותנו ממצב טקסט למצב גרפי. לפני כל פקודה גרפית אחרת אנו צריכים לקרוא לפקודה זו.

במסמך זה לא אתרכז בפרמטרים השונים שניתן להעביר לפונקציה. נציג את דרך הקריאה הסטנדרטית לפונקציה זו:

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>

int main()
{
      /* request auto detection */
      int gdriver = DETECT, gmode, errorcode;

      /* initialize graphics mode */
      initgraph(&gdriver, &gmode, "");

      /* read result of initialization */
      errorcode = graphresult();

      if (errorcode != grOk)  /* an error occurred */
      {
            printf("Graphics error: %s\n", grapherrormsg(errorcode));
            printf("Press any key to halt:");
            getch();
            exit(1);
      }

      ...
}
```

הפונקציה initgraph() מעבירה אותנו למצב גרפי.

graphresult() בודקת האם עברנו למצב הגרפי בהצלחה. במידה ולא, אנחנו מדפיסים הודעת שגיאה ומסיימים את התוכנית.

יציאה ממצב גרפי – closegraph()

בסוף העבודה במצב גרפי אנחנו צריכים לקרוא לפונקציה closegraph() כדי לחזור למצב טקסט.

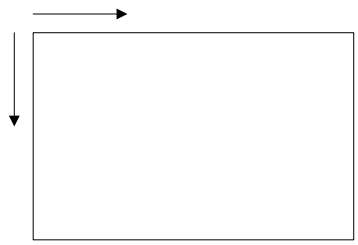
דוגמא לקריאה:

```
closegraph();
```

קבלת גודל המסך – getmaxx(), getmaxy()

כאשר אנחנו במצב גרפי, אנחנו מתייחסים למסך כאל מטריצה גדולה של נקודות. הנקודה השמאלית העליונה היא הנקודה $(0,0)$ והנקודה הימנית התחתונה היא $(\max X, \max Y)$. מכיוון שהרזולוציה של המסך עלולה להיות שונה בריצות שונות של התוכנית, הפקודות getmaxx() ו-getmaxy() מאפשרות לנו למצוא את הערכים המקסימליים עבור x ו-y, בהתאמה.

השרטוט הבא מצורף להבהרת צורת העבודה עם המסך במצב גרפי:



ציור קו – line()

ציור קו נעשה על ידי הפונקציה line, המקבלת שתי קורדינטות: קורדינטת התחלה וקורדינטת סיום.

```
line(x1, y1, x2, y2);
```

הפונקציה מציירת קו ישר בין שתי נקודות אלו.

צבע הציור – setcolor()

פונקציה זו מקבלת צבע אחד מהבאים: BLACK, BLUE, GREEN, CYAN, RED, MAGENTA, BROWN, LIGHTGRAY, DARKGRAY, LIGHTBLUE, LIGHTGREEN, LIGHTCYAN, LIGHTRED, LIGHTMAGENTA, YELLOW, WHITE ומשנה את צבע הציור הנוכחי לצבע זה.

לכל צבע יש גם ערך מספרי בו ניתן להשתמש.

כתיבת טקסט – outtextxy()

פונקציה זו מאפשרת לנו להדפיס תווים על המסך. היא מקבלת מקום – קורדינטת (x, y), ומחרוזת, ומדפיסה את המחרוזת במקום המבוקש.

```
outtextxy(x, y, "text");
```

שרטוט מעגל – circle()

הפונקציה מקבלת את קורדינטת מרכז העיגול, ואת הרדיוס שלו, ומציירת עיגול כמבוקש.

```
circle(x, y, radius);
```

נציג כעת תוכנית שלמה המצירת קו על המסך, המשלבת את הפונקציות השונות.

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>

int main()
{
      /* request auto detection */
      int gdriver = DETECT, gmode, errorcode;
      int xmax, ymax;

      /* initialize graphics and local variables */
      initgraph(&gdriver, &gmode, "");

      /* read result of initialization */
      errorcode = graphresult();
      /* an error occurred */
      if (errorcode != grOk)
      {
            printf("Graphics error: %s\n", grapherrormsg(errorcode));
            printf("Press any key to halt:");
            getch();
            exit(1);
      }

      setcolor(WHITE);
      xmax = getmaxx();
      ymax = getmaxy();

      /* draw a diagonal line */
      line(0, 0, xmax, ymax);

      /* clean up */
      getch();
      closegraph();
      return 0;
}
```

התוכנית נכנסת למצב גרפי, מציירת קו אלכסוני על המסך, מחכה שהמשתמש ילחץ על מקש כלשהו ומסתיימת.

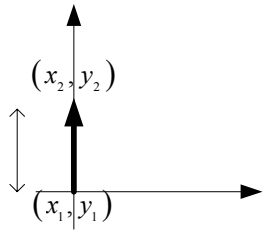
קצת מתמטיקה...

המטרה שלנו: לצייר שעון מחוגים על המסך.

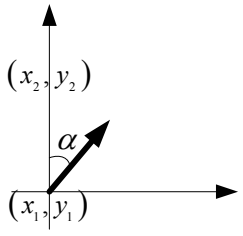
כדי להבין כיצד נעשה זאת, נעבור מספר שלבים.

נתחיל בחישוב מתמטי. אנו רוצים לצייר מחוג (קו) בודד המסתובב סביב ציר קבוע.

הנקודה (x_1, y_1) היא קבועה, ואילו (x_2, y_2) זהה עם כיוון השעון. האורך של הקו הינו קבוע - L.



שרטוט 1



שרטוט 2

נשים לב שב-C, הקורדינטות הממוקמות למעלה יותר הן הנמוכות יותר – הפוך מהמקובל בשרטוט גרפים. הקו שנצייר יתחיל כמשורטט בשרטוט 1, והוא ינטה עם כיוון השעון, כמתואר בשרטוט 2.

איך נחשב את (x_2, y_2) כתלות בזוית α? מהסתכלות בשרטוטים, נקבל:

$$x_2 = x_1 + \sin(\alpha) \cdot L$$
$$y_2 = y_1 - \cos(\alpha) \cdot L$$

מתחילים בכתיבת השעון!

תוכנית ראשונה – מחוג מסתובב

התוכנית הראשונה שנבנה תציג כיצד אנחנו מסובבים מחוג בודד.

בכל פעם נסובב אותו במעלה בודדת.

לצורך הדגמה נבחר מספר סיבובים כלשהו, למשל 3. המחוג יסתובב שלושה סיבובים, ולאחר מכן התוכנית תסתיים.

הלולאה המרכזית תהיה על הזוית α (נשים לב שהזוית מחושבת ברדיאנים):

```
for (angle = 0; angle < 2 * PI * CIRCLES_NUMBER; angle += (2 * PI / 360))
{
      ...
}
```

בכל פעם נצייר קו אשר הזוית שלו מהקו המתואר בשרטוט 1 היא angle.

להלן קוד התוכנית:

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <math.h>

#define PI 3.141592
#define L 50
#define CIRCLES_NUMBER 3

int main(void)
{
      int gdriver = DETECT, gmode, errorcode;
      int xmax, ymax;
      double angle;
      int x1, y1, x2, y2;

      initgraph(&gdriver, &gmode, "");

      errorcode = graphresult();
      if (errorcode != grOk)
      {
            printf("Graphics error: %s\n", grapherrormsg(errorcode));
            printf("Press any key to halt:");
            getch();
```

```
            exit(1);
      }

      x1 = getmaxx() / 2;
      y1 = getmaxy() / 2;
      x2 = x1;
      y2 = y1 - L;

      setcolor(WHITE);
      line(x1, y1, x2, y2);
      for (angle = 0; angle < 2 * PI * CIRCLES_NUMBER; angle += (2 *
PI / 360))
      {
            delay(10);
            setcolor(BLACK);
            line(x1, y1, x2, y2);

            x2 = x1 + sin(angle)*L;
            y2 = y1 - cos(angle)*L;
            setcolor(WHITE);
            line(x1, y1, x2, y2);
      }

      /* clean up */
      getch();
      closegraph();
      return 0;
}
```

קבענו את המיקום של (x_1, x_2) בדיוק למרכז המסך, ואת אורך הקו ל-50 נקודות. בכל פעם אנחנו מציירים קו לבן, מחכים מאית שניה, מוחקים אותו על ידי קו שחור, ומציירים קו חדש בזוית חדשה.

על ידי החלפת השורות

```
      x2 = x1 + sin(angle)*L;
      y2 = y1 - cos(angle)*L;
```

בשורות

```
      x2 = x1 + sin(angle)*(L + sin(angle) * 20);
      y2 = y1 - cos(angle)*(L + sin(angle) * 20);
```

נוכל לקבל אפקט יפה נוסף: המחוג יקטן ויגדל לחילופין, בהתאם לכיוון אליו הוא מצביע. תיווצר אשליה כאילו מהירות המחוג משתנה בעוד שבפועל היא זהה כל הזמן.

תוכנית שניה – מחוג המסתובב בדילוגים של שניות, שיפור ממשק

נתקדם שלב נוסף. נשנה את התוכנית ככה שהמחוג יזוז כל עוד לא נלחץ מקש. כאשר יילחץ מקש התוכנית תסתיים. כמו כן, במקום להתקדם כל פעם במעלה, נתקדם ב-6 מעלות, ונחלק את הסיבוב ל-60 צעדים.

הפונקציה kbhit() מחזירה אמת אם נלחץ מקש על המקלדת, או שקר (0) אם לא נלחץ מקש.
במידה ונלחץ מקש, הפונקציה getchar() או getch() תקרא אותו.
בלולאת ה-while בתוכנית הבאה אנחנו למעשה אומרים "כל עוד לא נלחץ מקש, בצע".

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <math.h>

#define PI 3.141592
#define L 50

int main(void)
{
   int gdriver = DETECT, gmode, errorcode;
   int xmax, ymax;
   double angle;
   int x1, y1, x2, y2;

   initgraph(&gdriver, &gmode, "");

   errorcode = graphresult();
   if (errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
   }

   x1 = getmaxx() / 2;
   y1 = getmaxy() / 2;
   x2 = x1;
   y2 = y1 - L;

   setcolor(WHITE);
   line(x1, y1, x2, y2);

   angle = 0;
   while (!kbhit())
   {
      delay(300);
```

```
        setcolor(BLACK);
        line(x1, y1, x2, y2);

        x2 = x1 + sin(angle)*L;
        y2 = y1 - cos(angle)*L;
        setcolor(WHITE);
        line(x1, y1, x2, y2);
        angle += (2 * PI / 360) * 6;
    }



    /* clean up */
    getch();
    closegraph();
    return 0;
}
```

תוכנית 3 - השעון

כעת נעבור לכתיבת השעון. ניקח את התוכנית הקודמת, ונרחיב אותה כך שתתאים למטרה שלנו.

כעת אנחנו צריכים שני מחוגים – אחד עבור הדקות, ואחד עבור השניות.

נשכפל את המשתנים של המחוג הבודד, וכעת מחוג השניות יסתובב בדיוק כמו בדוגמא הקודמת, ובנוסף יהיה לנו מחוג דקות לו נשמור זווית משלו ואורך משלו.

כמו כן, נשמור משתנה שיגדל אחרי כל תזוזה של מחוג השניות. פעם ב-60 תזוזות, נזיז את מחוג הדקות.

בדוגמא: SEC_LEN הוא אורך מחוג השניות, MIN_LEN הוא אורך מחוג הדקות, sec_x1, sec_y1, sec_x2, sec_y2 מציינים את הקורדינטות של מחוג השניות, ו- min_x1, min_y1, min_x2, min_y2 מציינים את הקורדינטות של מחוג הדקות.

שני המחוגים מסתובבים כאשר צירם הוא מרכז המסך.

בנוסף אנחנו משרטטים עיגול מסביב למחוגים, וכן רושמים את מספרי השעות, כדי ליצור שעון אמיתי.

קוד התוכנית:

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <math.h>

#define PI 3.141592
#define SEC_LEN 70
#define MIN_LEN 50

int main(void)
{
   int gdriver = DETECT, gmode, errorcode;
   int xmax, ymax;
   double sec_angle, min_angle;
   int sec_x1, sec_y1, sec_x2, sec_y2;
   int min_x1, min_y1, min_x2, min_y2;
   int time;

   initgraph(&gdriver, &gmode, "");

   errorcode = graphresult();
   if (errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
   }

   sec_x1 = getmaxx() / 2;
   sec_y1 = getmaxy() / 2;
   sec_x2 = sec_x1;
   sec_y2 = sec_y1 - SEC_LEN;

   min_x1 = getmaxx() / 2;
   min_y1 = getmaxy() / 2;
   min_x2 = min_x1;
   min_y2 = min_y1 - MIN_LEN;

   outtextxy(sec_x1 - 8, sec_y1 - SEC_LEN - 15, "12");
   outtextxy(sec_x1 + SEC_LEN + 10, sec_y1, "3");
   outtextxy(sec_x1 - 2, sec_y1 + SEC_LEN + 10, "6");
   outtextxy(sec_x1 - SEC_LEN - 15, sec_y1, "9");
   circle(sec_x1, sec_y1, SEC_LEN+25);

   setcolor(WHITE);
   line(sec_x1, sec_y1, sec_x2, sec_y2);

   sec_angle = 0;
   min_angle = 0;
   time = 0;
   while (!kbhit())
   {
```

```
        delay(200);
        setcolor(BLACK);
        line(sec_x1, sec_y1, sec_x2, sec_y2);
        line(min_x1, min_y1, min_x2, min_y2);

        sec_x2 = sec_x1 + sin(sec_angle)*SEC_LEN;
        sec_y2 = sec_y1 - cos(sec_angle)*SEC_LEN;

        min_x2 = min_x1 + sin(min_angle)*MIN_LEN;
        min_y2 = min_y1 - cos(min_angle)*MIN_LEN;


        setcolor(BLUE);
        line(sec_x1, sec_y1, sec_x2, sec_y2);
        setcolor(YELLOW);
        line(min_x1, min_y1, min_x2, min_y2);

        sec_angle += (2 * PI / 360) * 6;

        if (++time % 60 == 0) min_angle += (2 * PI / 360) * 6;
    }



    /* clean up */
    getch();
    closegraph();
    return 0;
}
```

EOF