Computer graphics (course project : solar system)

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


 /* manipulates the position of planets on the orbit */
 void planetMotion(int xrad, int yrad, int midx, int midy, int x[60], int y[60]) {
      int i, j = 0;

      /* positions of planets in their corresponding orbits */
      for (i = 360; i > 0; i = i - 6) {
            x[j] = midx - (xrad * cos((i * 3.14) / 180));
            y[j++] = midy - (yrad * sin((i * 3.14) / 180));
      }
      return;
 }

 int main() {
      /* request auto detection */
      int gdriver = DETECT, gmode, err;
      int i = 0, midx, midy;
      int xrad[9], yrad[9], x[9][60], y[9][60];
      int pos[9], planet[9], tmp;

      /* initialize graphic mode */
      initgraph(&gdriver, &gmode, "C:/TURBOC3/BGI");
      err = graphresult();

      if (err != grOk) {
            /* error occurred */
            printf("Graphics Error: %s",
                        grapherrormsg(err));
            return 0;
      }

      /* mid positions at x and y-axis */
      midx = getmaxx() / 2;
      midy = getmaxy() / 2;

      /* manipulating radius of all 9 planets */
      planet[0] = 7;
      for (i = 1; i < 9; i++) {
```

```
        planet[i] = planet[i - 1] + 1;
}

/* offset position for the planets on their corresponding orbit */
for (i = 0; i < 9; i++) {
        pos[i] = i * 6;
}

/* orbits for all 9 planets */
xrad[0] = 60, yrad[0] = 30;
for (i = 1; i < 9; i++) {
        xrad[i] = xrad[i - 1] + 30;
        yrad[i] = yrad[i - 1] + 15;
}

/* positions of planets on their corresponding orbits */
for (i = 0; i < 9; i++) {
        planetMotion(xrad[i], yrad[i], midx, midy, x[i], y[i]);
}

while (!kbhit()) {
        /* drawing 9 orbits */
        setcolor(WHITE);
        for (i = 0; i < 9; i++) {
                ellipse(midx, midy, 0, 360, xrad[i], yrad[i]);
        }

        /* sun at the mid of the solar system */
                outtextxy(midx, midy, "    SUN");
                setcolor(YELLOW);
                setfillstyle(SOLID_FILL, YELLOW);
                circle(midx, midy, 20);
                floodfill(midx, midy, YELLOW);

                /* mercury in first orbit */
                setcolor(CYAN);
                setfillstyle(SOLID_FILL, CYAN);
                outtextxy(x[0][pos[0]],y[0][pos[0]], " MERCURY");
                pieslice(x[0][pos[0]], y[0][pos[0]], 0, 360, planet[0]);


                /* venus in second orbit */
                setcolor(GREEN);
                setfillstyle(SOLID_FILL, GREEN);
                outtextxy(x[1][pos[1]],y[1][pos[1]], " VENUS");
                pieslice(x[1][pos[1]], y[1][pos[1]], 0, 360, planet[1]);

                /* earth in third orbit */
```

```
setcolor(BLUE);
setfillstyle(SOLID_FILL, BLUE);
outtextxy(x[2][pos[2]],y[2][pos[2]],  " EARTH");
pieslice(x[2][pos[2]], y[2][pos[2]], 0, 360, planet[2]);

/* mars in fourth orbit */
setcolor(RED);
setfillstyle(SOLID_FILL, RED);
outtextxy(x[3][pos[3]], y[3][pos[3]], " MARS");
pieslice(x[3][pos[3]], y[3][pos[3]], 0, 360, planet[3]);

/* jupiter in fifth orbit */
setcolor(BROWN);
setfillstyle(SOLID_FILL, BROWN);
outtextxy(x[4][pos[4]],y[4][pos[4]],  " JUPITER");
pieslice(x[4][pos[4]], y[4][pos[4]], 0, 360, planet[4]);

/* saturn in sixth orbit */
setcolor(LIGHTGRAY);
setfillstyle(SOLID_FILL, LIGHTGRAY);
outtextxy(x[5][pos[5]], y[5][pos[5]], "  SATURN");
pieslice(x[5][pos[5]], y[5][pos[5]], 0, 360, planet[5]);

/* uranus in sevth orbit */
setcolor(BROWN);
setfillstyle(SOLID_FILL, BROWN);
outtextxy (x [6] [pos [6]], y [6] [pos [6]], "  URANUS");
pieslice(x[6][pos[6]], y[6][pos[6]], 0, 360, planet[6]);

/* neptune in eigth orbit */
setcolor(LIGHTBLUE);
setfillstyle(SOLID_FILL, LIGHTBLUE);
outtextxy (x [7] [pos [7]],y [7] [pos [7]],  "  NEPTUNE");
pieslice(x[7][pos[7]], y[7][pos[7]], 0, 360, planet[7]);

/* pluto in ninth orbit */
setcolor(LIGHTRED);
setfillstyle(SOLID_FILL, LIGHTRED);
outtextxy (x [8] [pos [8]],y [8] [pos [8]],  "  PLUTO");
pieslice(x[8][pos[8]], y[8][pos[8]], 0, 360, planet[8]);

/* checking for one complete rotation */
for (i = 0; i < 9; i++) {
        if (pos[i] <= 0) {
                pos[i] = 59;
        } else {
                pos[i] = pos[i] - 1;
        }
```

```
            }

            /* sleep for 100 milliseconds */
            delay(100);

            /* clears graphic screen */
            cleardevice();
        }

        /* deallocate memory allocated for graphic screen */
        closegraph();
    return 0;
}
```

Output: