I. gyakorlat - Jegyzőkönyv

Ekart Csaba - ZWPMKP

1. feladat

Az első feladatot még az órán közösen elkészítettük.

2. feladat

A feladat megoldásához egy structot készítettem, amelyben tárolható minden egyes pont x, illetve y koordinátája. Ezzel a point típussal létrehozott szükséges egyik global változóm egy pont típusú elemeket tároló vektor volt, melyben a “lerakott” pontokat tárolom, illete egy boolean, amely a szaggatottságot szabályozza.

A feladatnak megfelelően minden egyes bal egérnyomásra egy pontot veszek fel az egér koordinátáin, melyet egyből hozzáadok a vektoromhoz.
A vonalat a GL_LINE_STRIP és a glVertex2f parancsokkal rajzoltam ki.

```
60      glBegin(GL_LINE_STRIP);
61      for(int i = 0; i < points.size(); i++) {
62          glVertex2f(points[i].x, points[i].y);
63      }
64
65      glEnd();
```

A vonal szaggatottságához egy külön logikai változót hoztam létre, mely alapvető értéke false, de igazzá változik az ‘s’ megnyomás hatására, és logikai állapotától függően szaggatottá állítja a vonalat. A vonal vastagság állításához ugyanabban a switch-case-ben helyeztem el a szükséges elemeket.

3. feladat

A feladat során az előzőeket kibővítettem egy global integerrel, amelynek az értéke alapból -1, ilyen index ugyenis nem szerepelhet a pontok vektorában. Jobb egérgomb lenyomásakor érzékeljük, hogy a lent lévő pontok közül melyikhez nyomtunk le viszonylag közel, ezt innentől kezdve kiválasztottnak tekintjük. A kiválasztott indexen lévő elem koordinátája a motion függvényben folyton változik. A jobb gomb felengedése után ismét érvénytelenek lesz az index, így egyik elem se “mozgolódik”.

```
    //jobb gomb lenyomására/felengedésére (lásd 2.2 feladat)
    if (button == GLUT_RIGHT_BUTTON) {
        if  (state == GLUT_DOWN) {
            int radius = 5;
            for(int i = 0; i < points.size(); i++) {
                if (((points[i].x - radius < x) && (x < points[i].x + radius)) &&
                    ((points[i].y - radius < y) && (y < points[i].y + radius))) {
                    //debug
                    //std::cout << "PONT KOORDI: " << points[i].x << " " << points[i]
                    //points[i].set_coord(x,y);
                    selected_point = i;
                    break;
                }
            }
        } else {
            selected_point = -1;
        }
```

```
//Egér mozgára aktiválódik, kurzorpozíció koordinátáit kapja bemenetnek
void Motion(int x, int y) {
    if (selected_point != -1) {
        points[selected_point].set_coord(x,y);
    }
```

4. feladat

Az előző feladatban elért eredményeket bővítettem. Egyetlen pont vektor helyett egy vonalak vektor van, amelynek az elemei a fentiekhez hasonló pont vektorok (mivel minden egyes szakasz pontok halmaza).

Az ábrázolásnál szükséges módosítás volt, hogy minden egyes vonalat külön ábrázoljunk, és a glBegin és glEnd elhelyezése is kulcsfontosságú, hogy a szakaszok egymástól egyértelmáen elkülönüljenek.

```
    for (int j = 0; j < lines.size(); j++) {
        glBegin(GL_LINE_STRIP);
        if(lines[j].size()>1) {
            for(int i = 0; i < lines[j].size(); i++) {
                glColor3b(0,255,0);
                glVertex2f(lines[j][i].x, lines[j][i].y);
            }
        }
        glEnd();
    }
```

Egy ‘a’ nevű golbal változóban az éppen lentlévő szakaszok számát tároltam, hogy tudjam éppen melyik szorul “bővítésre”, illetve egy logikai változóban tároltam az egér lenyomásának állapotát.

Kattintásra a logikai változó igazzá változik, és egy új vonal kezdetét veszi. Ekkor egér mozgáskor új pontot veszünk fel, mellyel a jelenlegi vonalat “bővítem”, így a mozgatás hatására új és új pontok kerülnek a vektorba. Az egér gomb felengedésekor a logikai változó hamissá válik, és a következő vonalra ugrunk.

```
void Motion(int x, int y) {
    if (pushed) {
        point newPoint(x,y); // Make a new segment
        lines[a].push_back(newPoint); // Insert segment into segment vector
    }
    glutPostRedisplay();
}
```

```
void MouseFunction(int button, int state, int x,int y)
{
    //bal gomb lenyomására
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN /*&& ... */) {
        lines.push_back(points);
        pushed = true;
    }

    if (button == GLUT_LEFT_BUTTON && state == GLUT_UP /*&& ... */) {
        pushed = false;
        a++;
    }
```

5. feladat

A menü elkészítéséhez a megadott példában írtam át a megfelelő értékeket.

A menügombok akcióinak kezelésére létrehozott függvényt némileg átírtam, úgy hogy amennyiben a 4-es értéket kapjuk (tehát a kilépésre kattint a felhasználó) megszakítjuk a program működését és kilépünk, ha pedig bármi másra, akkor a megfelelő menü kódot egy global változóban tárolom, amely alapján a redraw függvényben döntök hogy épp mit kell kirajzolnom. A kirajzoláshoz szükséges függvények a feladat leírásban szerepeltek.

```
    switch(melyik) {
        case 0:
            glutSolidTeapot(0.5);
            break;
        case 1:
            glutWireTeapot(0.5);
            break;
        case 2:
            glutSolidCube(0.5);
            break;
        case 3:
            glutWireCube(0.5);
            break;
    }
```