Számítógépes grafika alapjai

Szegedi Tudományegyetem
Informatikai Tanszékcsoport
Képfeldolgozás és Számítógépes Grafika Tanszék

2013-2014. tanév

A kurzusról

- Előadó és gyakorlatvezető: Németh Gábor
- Előadás (nappali tagozaton): heti 2 óra
- Gyakorlat (nappali tagozaton): heti 1 óra
- Követelmények:
  - 1 kötelező program a meghirdetett témakörben
  - 2 ZH (gép előtt)

Pontok rajzolása

Bevezetés történeti áttekintés

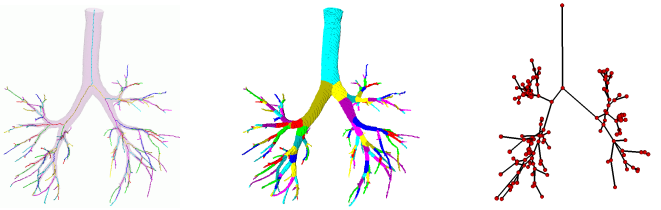
Bevezetés

Számítógépes grafika: Valós és képzeletbeli objektumok (pl. tárgyak képei, függvények) *szintézise* számítógépes modelljeikből (pl. pontok, élek, lapok).

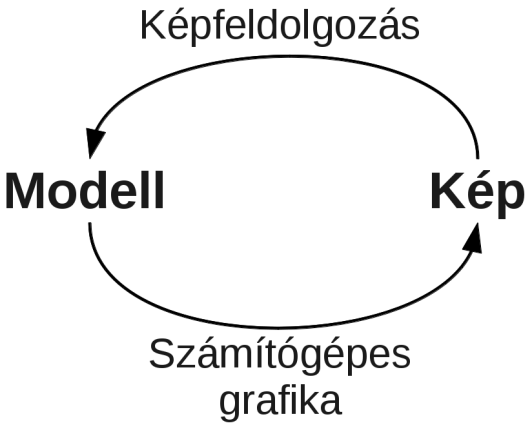
A számítógépes grafika feladata, hogy a tárgyak, objektumok számítógépes modelljeiből egy képet, látványt hozzon létre. A számítógépes modellben csupán a pontok, élek, lapok vannak meghatározva. Továbbá a modellben definált lehet egy vagy több fényforrás, a fények iránya és erőssége. A számítógépes grafika feladata az, hogy ezeknél az objektumot alkotó elemeknél kiszámolja a látható felületeket, az árnyékokat, a visszaverődött fényt, a felületek színét.

Bevezetés

Számítógépes képfeldolgozás: Képek analízise, objektumok modelljeinek rekonstrukciója képeikből (pl. légi-, űr-, orvosi felvételek kiértékelése, torzított képek helyreállítása).



A képfeldolgozás – a számítógépes grafikával ellentétben – a képből látványból alkotja meg a modellt. A képekből jellemzőket (szín, textúra, alak) határozunk meg. Ezek a jellemzők, egymáshoz hasonló képek esetén hasonló értékeket vesznek fel. Ezen értékeket osztályozva, csoportosítva, meghatározzuk a képen látható objektum modelljét.
A fenti ábrán egy háromdimenziós orvosi képből szegmentált légút látható. A légútat középvonalval jellemezzük, és az egyes ágakat más-más színnel címkézzük. A címkézett középvonalnak köszönhetően a légútfából gráfot készíthetünk, ahol a gráf csúcsait a középvonal elágazási pontjai és a végpontjai alkotják, az éleket pedig a színezett görbeszegmensekből kapjuk.

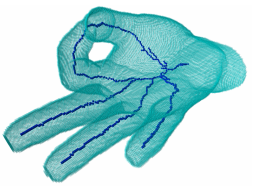

Képfeldolgozás
Modell
Kép
Számítógépes
grafika

Bevezetés

Tapasztalat, hogy képek formájában az adatok gyorsabban és hatásosabban feldolgozhatók az ember számára.

Fejlődés:

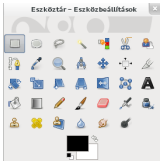
fotózás ⟶ televízió ⟶ számítógépes grafika



http://www.thenorthernecho.co.uk

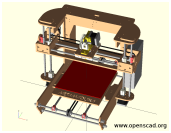
A számítógépes grafika kialakulásához és elterjedéséhez nagyban hozzájárult, hogy az emberi agy képek formájában gyorsabban és hatékonyabban dolgozza fel az információt, mint például szöveges úton, vagy beszéd útján.
Ahogy mondani szokták egy kép többet mond ezer szónál. Kezdetben a rajzokkal, illusztrációkkal, fotókkal muttak be eseményeket. Gondoljunk például a tankönyvek ábráira. El tudjuk képzelni a folyamatot, eseményeket. Illusztrálva van például egy óraszerkezet, le van rajzolva, vagy le van fényképezve egy kémiai kísérlet. Baleset esetén fotókkal dokumentálták az szerencsétlenség következményeit.
A televízió megjelenésével nagy tömegekhez lehet közvetíteni mozgó képet. Egy esemény tudósítása filmen sokkal inkább bemutatható, mint mondjuk fotókkal.
A számítógépes grafika megjelenésével nem csak felvételekről közölhetünk információt, hanem bemutathatunk szimulációkat is, valamint lehetőség nyílik az interaktív tanulásra is (pl. virtuális valóság). Bemutathatunk olyan eseményeket, amelyeket a való világban nem tudtunk dokumentálni rögzíteni.

Alkalmazási területek



felhasználói programok

üzlet, tudomány



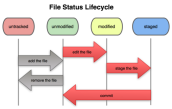
tervezés, CAD

torcs.sourceforge.net

játék, szimuláció

technology.desktopnexus.com

művészet, kereskedelem



git-scm.com

folyamatirányítás

térképészet

A számítógépes grafika ma már az élet nagyon sok területén jelen van, nem csak az informatikában. Az üzleti életben, vállalati közegben, ha előadást tart valaki általában használ hozzá prezentációt. Az prezentációban használhat animációkat, áttűnéseket, a design elemeket is a számítógépes grafika eszközeivel alakítja ki.

Számítógépes programoknál ma már alig elképzelhető, hogy ne készüljön a programhoz grafikus felhasználói felület, amelyen a felhasználó kényelmesebben kezelheti a programot.

Egyre több játék rendelkezik 3D grafikával, a megjelenítés egyre valósághűbb, de ezt a technológiát nem csak a játékok, de a tudományos modellező programok is használják. Ne feledkezzünk meg a számítógéppel támogatott tervezésről (Computer Aided Design, CAD) sem, hiszen ezen a területen is fontos szerepet tölt be a számítógépes grafika.

Történeti áttekintés

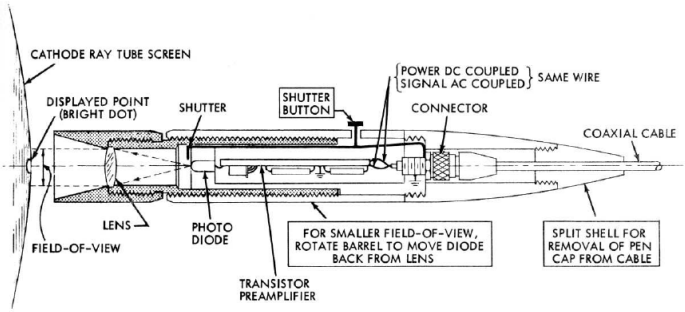
Kezdetben a képek megjelenítése teletype nyomtatókon történt.

1950:

- MIT: számítógéppel vezérelt képernyő
- SAGE légvédelmi rendszer (a programok képernyőről történő vezérlése fényceruzával)

A képeket kezdetben teletype nyomtatókkal jelenítették meg. Ezek a nyomtatók (hagyományos és speciális) karaktereket voltak képesek nyomtatni, így a képet úgy kellett megalkotni, hogy valóban kép formája legyen. Manapság újra kezd elterjedni az ún. „RTTY art”, vagyis amikor különböző karakterekből állítanak elő képeket annak megfelelően, hogy a karakter mekkora területet „fest be”. Ez a fajta megjelenítés a képfeldolgozásból ismert féltónusú (halftone) képek egy korai változatának tekinthető.
1950-ben az MIT elkészített egy számítógéppel vezérelt képernyőt. A SAGE (Semi-Automatic Ground Environment) nevű légvédelmi rendszer kifejlesztése Jay Forrester és George Valley nevéhez fűződik. Az operátorok egy fényceruza segítségével jelölték meg a képernyőn szereplő célpontot.

Történeti áttekintés



A fényceruza hegyében egy fényérzékeny szenzor található. A katódsugárcső megadott képfrissítési frekvenciával „rajzol” a képernyőre, és amikor a fényceruza érzékeli a képernyőn megjelenített képpontot (fényt), akkor egy elektromos jelet küld a számítógépnek. A számítógép ebből a jelből és a képernyőre irányított elekronsugár irányából számolja ki a fényceruza által mutatott abszolút pozíciót.

Történeti áttekintés

1963:

- A modern interaktív grafika megjelenése.
- I. Sutherland: Sketchpad
  Adatstruktúrák, szimbólikus struktúrák tárolása, interaktív megjelenítés, választás, rajzolás

1963-ban megjelenik az első interaktív grafikai program, az Ivan Sutherland által kifejlesztett Sketchpad. Ez a program tekinthető a mai CAD programok első változatának. Ivan Sutherland munkájának hatására fejlesztette ki a Xerox az első grafikus felhasználói felületet.

Történeti áttekintés

1964:

- CAD - DAC-1 (IBM)
- Autók tervezése (General Motors)

A DAC-1 (Design Augmented by Computer) volt az első valódi tervezőrendszer, amelyet a General Motors fejlesztett ki.

Történeti áttekintés

Lassú a fejlődés, mert

- drága a hardver
- drága számítógépes erőforrások
  (nagy adatbázis, interaktív manipuláció, intenzív adatfeldolgozás)
- nehéz volt nagy programokat írni
- a szoftver nem hordozható

A számítógépek a 60-as és 70-es években még nagyon nagy méretűek voltak, valamint csak a legnagyobb intézmények, vállalatok engedhették meg maguknak ezek üzemeltetését. Az egyes számítógéptípusokra más-más programozási nyelven kellett programokat írni, emiatt a szoftver nem volt hordozható.

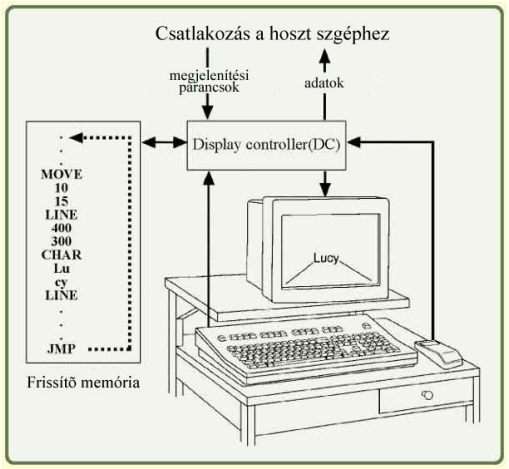
Történeti áttekintés

1960-as évek:
Jellemző output eszköz az ún. **vektor-képernyő** (szakaszokat rajzol ponttól pontig)

Részei:

- Képernyő processzor (DP) - mint I/O periféria kapcsolódik a központi egységhez
- Képernyő tároló memória – a megjelenítéshez szükséges program és adat tárolására
- Képernyő - katód sugár cső

Történeti áttekintés

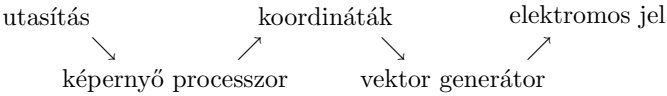


A korai grafikai rendszerek a vektor-képernyő adottságait használták ki. Egyszerű utasításokkal rajzoltak ki szakaszokat, köröket, köríveket, görbéket, szöveget. Az ábrán látható példa egy egyszerű rajzot mutat be.

1. Menj a (10,15)-ös koordinátára.
2. Rajzolj szakaszt a (400, 300)-as koordinátáig.
3. Írj ki két karaktert: Lu *(megjegyzés: 16 bit)*
4. Írj ki két karaktert: cy
5. Rajzolj szakaszt ...
6. ...
7. Menj vissza a program elejére

Az utolsó lépés azért fontos, mert a katódsugárcsöves képrnyőkön a kép folyamatos frissítések árán maradt látható.

Történeti áttekintés



30 Hz-es frissítés (foszforeszkáló képernyő - nem villog annyira)

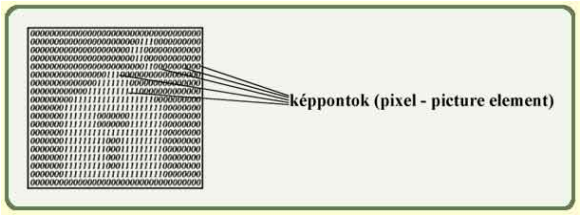
1960-as évek vége:
DVST (direct-view storage cube) - a látványt közvetlenül tároló cső: olcsóbb

képernyő $\longleftrightarrow$ kisszámítógép
felszabadul a központi gép

Történeti áttekintés

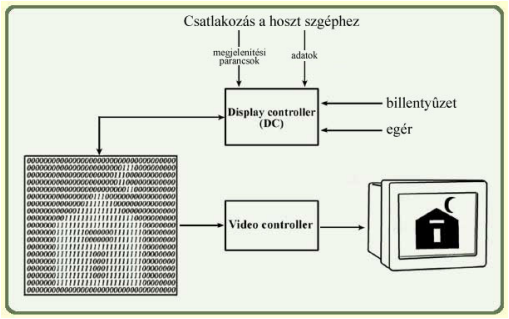
1968:
A hardver képes a skálát változtatni, a képet mozgatni, vetületeket előállítani valós időben.

1970-es évek:
Jellemző output eszköz a **raszter képernyő**
(TV-technika, bit-térképes grafika)
Bit-térkép (bitmap)
képek reprezentálása bitmátrixszal



Az 1960-as évek végére a technika odáig fejlődött, hogy a vektorképernyőre kirajzolt görbéket skálázni (kicsinyíteni, nagyítani), valamint mozgatni is lehetett.
Az 1970-es években megjelentek a raszter képernyők, amelyek egy bittérképes grafikákat voltak képesek megjeleníteni. A bit-térkép nem más, mint egy olyan mátrix, amelynek celláiban csak 0-k és 1-esek lehetnek.

Történeti áttekintés



A raszteres képernyők a grafikus primitíveket (pixel - képpont) ún. frissítő tárolóban tartják.

A raszteres képernyőkön megjelenített kép egy köztes frissítő tárolóban tárolódik. A videovezérlő feladata, hogy a képernyőn azokat a képpontokat, ahol a tárolóban 1-es szerepel feketével, a 0-asakat pedig fehérrel jelenítse meg.
Felhasználói interakció hatására (egér, billentyűzet) a *Display controller* felülírja a tároló tartalmát, és a video vezérlő frissíti a képet.

Történeti áttekintés

mátrix – raszter sorok – képpontok

Bit-térképek:
$1024 \cdot 1024 \cdot 1 = 128$ k bináris kép

Pixel képek:
$1024 \cdot 1024 \cdot 8 = 256$ szürkeárnyalat vagy szín
$1024 \cdot 1024 \cdot 24 = 2^{24}$ szürkeárnyalat vagy szín

Ma tipikus:
$1080 \cdot 1024 \cdot 24 \approx 3{,}75$ MB RAM

Egy bitmátrix raszter sorokból áll, amelyket képpontokra lehet bontani. Nézzük meg a dián, mekkora adatmennyiséget kell eltárolni abban az esetben, ha a képponthoz tartozó adatot egy biten, egy bájton vagy 3 bájton ábrázoljuk.
Megjegyzem, hogy a pixel képek esetében a szürkeárnyalatos képek pixel-értékeit 8 biten ábrázoljuk (0-255-ig). Ritkán, főleg orvosi képeken előfordulhat, hogy ennél nagyobb intervallumra van szükség.
Amennyiben színes képeket szeretnénk megjeleníteni (és tárolni), úgy 8 bit esetén 256 különböző színt tudunk tárolni (indexelt képek), vagy pedig a vörös, a zöld, illetve a kék színcsatornához is rendelhetünk 8-8 bitet, vagyis csatornánként 256 árnyalatot.

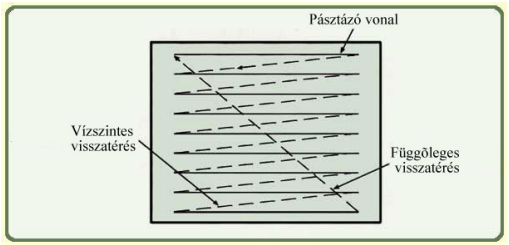
Történeti áttekintés

A raszteres képernyő előnyei:

- Olcsó logikájú processzor (soronként olvas).
- A területek színekkel kitölthetők.
- Az ábra bonyolultsága nem befolyásolja a megjelenítést.

A raszteres képernyő hátrányai:

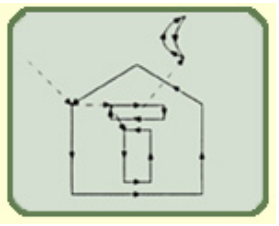
- A grafikus elemeket (pl. vonal, poligon) át kell konvertálni (RIP - Raster Image Processor)
- A geometriai transzformációk számításigényesek.



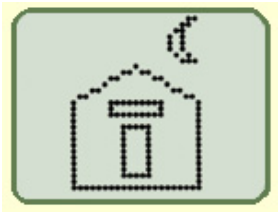
A raszteres képernyő történő kirajzolás mindig soronként történik. A pásztázás mindig balról jobbra és lentről felfelé halad.

Megjelenítés raszteres képernyőn

Ideális vonalrajz



Vektoros kép



Raszteres kép

Raszteres kép terület kitöltéssel

Figyeljük meg, hogyan jelenik meg a kirajzolt forma vektoros és raszteres képernyőn!
Az ideális vonalrajz egy házat és egy holdat ábrázol. A vektoros kép bemutatja, hogy hogyan kell a görbéket és a szakaszokat kirazjzolni a képernyőre. Az megjelenített eredmény nagyon hasonlít az ideális vonalrajzhoz. A raszteres kép esetében már kicsit „darabosabb” a megjelenített eredmény. A raszteres feldolgozás előnye viszont, hogy akár kitöltött területek is berajzolhatók.

Történeti áttekintés

1980-as évekig:
A számítógépes grafika szűk speciális terület a drága hardver miatt.

Újdonságok:

- Személyi számítógépek (Apple Machintosh, IBM PC)
- Raszteres képernyők
- Ablak technika

Eredmény:

- Sok alkalmazás
- Sok I/O eszköz (pl. egér, tábla)
- Kevesebbet használjuk a billentyűzetet (ikonok, menük, ...)

Történeti áttekintés (szoftverek)

Eszköz-függő $\xrightarrow{\text{fejlődés}}$ eszközfüggetlen
Így lehet „hordozható” a felhasználói szoftver

1977:
3D Core Graphics System

1985:
GKS (Graphics Kernel System) 2D

Történeti áttekintés (szoftverek)

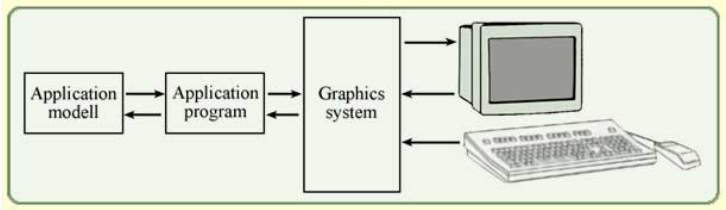
1988:

- GKS - 3D
- PHIGS (Programmer's Hierarchical Interactive Graphics System)
  - Logikailag kapcsolódó primitívek csoportosítása szegmensekbe,
  - 3D primitívek egymásba ágyazott hierarchiája,
  - Geometriai transzformációk,
  - Képernyő automatikus frissítése, ha az adatbázis változik

1992:

- OpenGL (SGI)

Interaktív grafikai rendszerek



Interaktivitás
A felhasználó vezérli az objektumok kiválasztását, megjelenítését billentyűzetről, vagy egérrel...

Az alkalmazás egy modellt tárol, amelyet a program a továbbít a grafikai rendszernek, amely kiszámolja, hogy a modellnek megfelelő látványt hogyan kell kirajzolni. A modell tartalmazhatja a fényforrásokat, azok pozícióját, jellemzőit, a kamera helyzetét és azt, hogy hová néz, valamint a „tárgyakat".
Ha a felhasználó valamilyen műveletet végez programban (például egérrel megfogja és elforgatja a kirajzolt modellt), akkor ez az interakció a grafikus rendszer továbbítja a programnak, amely áttranszformálja a modellt az interakciónak megfelelően, majd ha kész a transzformáció, a program továbbítja az aktuális modellt a grafikus rendszernek, hogy frissítse a képernyőn a korábbi látványt.

Felhasználói modell adatbázis:

- Adatok, objektumok, kapcsolatok (adattömb, hálózati adatok listája, relációs adatbázis)
- Primitívek (pontok, vonalak, felületek)
- Attribútumok (vonal stílus, szín, textúra)

Az adatokat relációs modellnek megfelelően listákban tároljuk. Egy-egy grafikus primitívnek meghatározhatjuk az attribútumait.

Interaktív grafikai rendszerek

Az interaktívitás kezelése:
Tipikus az esemény-vezérelt programhurok:

```
kezdeti képernyő beállítás;
while (true) {
  parancsok vagy objektumok választhatók;
  várakozás, amíg a felhasználó választ;
  switch (választás)
    case 'választott': a modell és a képernyő frissítésére; break;
     ...
     case (quit) exit(0);
}
```

A számítógépes grafika osztályozása I.

Dimenzió szerint:

- 2D
- 3D

Képfajta szerint:

- vonalas
- szürkeárnyalatos
- színes (árnyékolt)

A számítógépes grafika osztályozása II.

Interaktivitás szerint:

- Offline rajzolás
- Interaktív rajzolás (változó paraméterekkel)
- Objektum előre meghatározása és körüljárása
- Interaktív tervezés

Kép szerepe szerint:

- Végtermék
- Közbülső termék

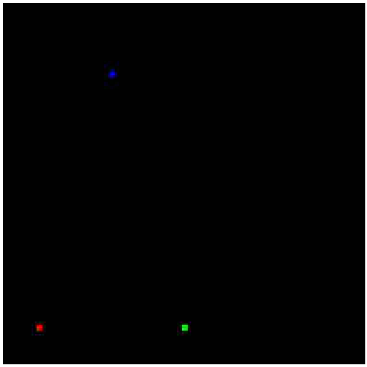
Az offline rajzolás azt jelenti, hogy a modellt valahogy megjelenítjük a képernyőn, és ezen már nem változtatunk, egy kész képet állítunk elő. Az interaktív rajzolás változó paraméterrek ezzel szemben azt takarja, az objektumaink nem változnak, de bizonyos paraméterek, például megvilágítás változhat.
Az objektum előre meghatározása körüljárásal azt a típusú megjelenítést takarja, amelynél a kamera vagy a teljes modell (a néző szemszögéhez képest) elmozdul.
Az interaktív tervezésre pedig jó példa a 3D Studio-ból ismert deformálás művelet, amikor megfogjuk a felszíni háló egy pontját, és ennek a pontnak a mozgatásával a háló többi része is változik.

OpenGL

Pontok rajzolása

Pontok rajzolása

Rajzoljunk egy piros pontot a (10, 10), egy zöld pontot az (50, 10) és egy kék pontot a (30, 80) koordinátákba (az ablak 100*100-as méretű)



A következőkben megnézünk egy példát, ahol három pontot rajzolunk ki 2-dimenzióban.

Színmodellek, színcsatornák

RGBA színmodell:
Minden színt négy komponens definiál (R, G, B, A)

- R - vörös (red)
- G - zöld (green)
- B - kék (blue)
- A - átlátszóság (alpha)

Minél nagyobb az RGB komponens értéke, annál intenzívebb a színkomponens.

Átlátszóság:

- 0.0: teljesen átlátszó
- 1.0: nem átlátszó

Például: (0.0, 0.0, 0.0, 0.0) – átlátszó fekete

Az OpenGL grafikus függvénykönyvtár az RGBA színmodellt használja. Minden színcsatorna alapértelmezésként a [0.0; 1.0] intervallumon veszi fel értékét (megjegyzem, hogy ez függ a paraméter típusától is, erről később lesz még szó). A 0.0 a legsötétebb, legkevésbé intenzív érték, még az 1.0 a legintenzívebb érték. Az A (alpha) csatorna felel az átlátszóságért. A vörös, a zöld és a kék színek az alapszínek az additív színkeverésnél. Az RGB színmodellel főképpen a megjelenítők, képernyők dolgoznak, a nyomdaiparban a CMY vagy CMYK színodellt (szubtraktív színkeverés) használják.

Törlő szín

Az aktuális törlőszín beállítására szolgáló függvény:

```
void glClearColor(
     GLclampf red,
     GLclampf green,
     GLclampf blue,
     GLclampf alpha)
```

Alapértelmezett érték: (0.0, 0.0, 0.0, 0.0);
`GLcampf` - `float`

A képernyő törléséhez (amely nagyon gyakran a képernyő frissítésekor következik be) a törlőszínt a `glClearColor` függvénnyel állíthatjuk be. Négy paramétereként a színcsatornák értékét kell megadni. A `GLcampf` típus az OpenGL-ben van definiálva, megfelel a C nyelvből ismert `float` típusnak.

Színbeállítás

A rajzoló (vagy kitöltő) szín beállítása:
`void glColor{34}{b s i f d ub us ui} (T components);`

- b - byte
- s - single
- i - integer
- f - float
- d - double
- u - unsigned

A színbelállítás a csúcspontokhoz van hozzárendelve.
Például:

- `glColor3f(1.0f, 0.0f, 0.0f)` – piros
- `glColor3f(0.0f, 1.0f, 0.0f)` – zöld
- `glColor3f(0.0f, 0.0f, 1.0f)` – kék

A színek beállítására a `glColor*` valamely változata szolgál. A kapcsos zárójelben szereplő számok illetve betűk valamelyik behelyettesíthető a függvény nevébe. A 3 vagy 4 szám azt jelzi, hogy három vagy négy komponenst szeretnénk megadni. A második kapcsos zárójelben felsorolt betűkódok pedig azt mutatják meg, hogy milyen adattípusú a megadott paraméter.

Csúcsopontok megadása

Csúcspont(ok) (vertexek) megadása:

```
void glVertex{234}{sifd}( T coords );
```

Például:

```
glVertex2i(20, 10) // a pont koordinátája (20,10)
```


A pontokat a `glVertex*` függvényekkel tudjuk megadni. A kapcsos zárójelben lévő szám a függvény nevében a dimenziót jelzi (és egyben a paraméterek számát is), az `i`, `f`, `d`, `s` betű pedig a paraméter típusát jelzi.

- `i` – `integer`
- `f` – `float`
- `d` – `double`
- `s` – `single`

Objektumok megadása

Az objektumok megadása a `glBegin(enum mode)` és `glEnd()` függvények között történik.

```
void glBegin(enum mode)
     ...
glEnd();
```

A `mode` értéke lehet pl.
`POINTS`, `LINES`, `POLYGON`

Az objektumok megadása a `glBegin()` ... `glEnd()` függvények között kell leírni az elemek csúcsait (vertexeit). A vonal- és poligonrajzolásról később még lesz szó.

Mátrix módok megadása

Transzformációk: mátrixokkal vannak definiálva

- nézeti (viewing),
- modellezési (modelling),
- vetítési (projection)

```
void glMatrixMode(enum mode)
```

Ha a `mode == GL_PROJECTION`, akkor a vetítési mátrix van érvényben. Például:

```
glMatrixMode(GL_PROJECTION)
```

`void glLoadIdentity(void)` az érvényes mátrix az egységmátrix lesz.

Az OpenGL-ben háromféle mátrix módot különböztetünk meg:

- nézeti: a látványt, kamera nézetet transzformálja,
- modell: a modellt transzformálja,
- vetítési: a vetítést transzformálja.

Vetítési mátrix megadása (2D)

Ortografikus vetítés:
`void gluOrtho2D(double left, double right, double bottom, double top)`

Ez a vetítés az objektum 2D merőleges vetítése adja meg a (left, right, bottom, top) téglalapra.

Például:

```
gluOrtho2D(0,100, 0, 100);
```


A `gluOrtho2D()` függvény az objektumokat a függvény paraméterei által definiált téglalapra vetíti merőlegesen. A példában szereplő téglalap mérete 100×100-as, és a vízszintes és függőleges tengelyhez illeszkedik.

Pufferek törlése

A pufferek tartalmának törlése:
A pufferek:

- `GL_COLOR_BUFFER_BIT`
- `GL_DEPTH_BUFFER_BIT`
- `GL_STENCIL_BUFFER_BIT`
- `GL_ACCUM_BUFFER_BIT`

Például: a szín buffer törlése

```
glClear(GL_COLOR_BUFFER_BIT)
```

Képernyő módok

A képernyőmódot a `glutInitDisplayMode(enum mode)` függvénnyel lehet beállítani.

Például: ha `mode == SINGLE | GLUT_RGB`, akkor az ún. egyszeresen pufferelt RGB módban specifikál ablakot.

Ablak

`void glutInitWindowSize(int width, int height)`: az ablak méretét definiálja.

`void glutInitWindowPosition(int x, int y)`: az ablak helyét (bal felső sarkának pozícióját) adja meg.

`int glutCreateWindow(char* name)`: Megnyit egy ablakot az előző rutinokban specifikált jellemzőkkel. Ha az ablakozó rendszer lehetővé teszi, akkor name megjelenik az ablak fejlécén. A visszatérési érték egy egész, amely az ablak azonosítója.

Callback függvények

`void glutDisplayFunc(void(*func)(void))`: Azt a callback függvényt specifikálja, amelyet akkor kell meghívni, ha az ablak tartalmát újra akarjuk rajzoltatni.
Pl.: `glutDisplayFunc(display);`

`void glutKeyboardFunc(void(*func)(unsigned char key, int x, int y))`: Azt a callback függvényt specifikálja, melyet egy billentyű lenyomásakor kell meghívni. A `key` egy ASCII karakter. Az `x` és `y` paraméterek az egér pozícióját jelzik a billentyű lenyomásakor (ablak relatív koordinátákban).
Pl.: `glutKeyboardFunc(keyboard);`

A példákban megadott `display()` és `keyboard()` függvényeket a programunk külön tartalmazni fogja. Ezekkel a callback függvényekkel azt mondjuk meg, hogy melyik függvény fogja feldolgozni az eseményt.

Pontrajzoló program I.

```
#include <GL/glut.h>

void init(void) {
   glClearColor(0.0, 0.0, 0.0, 0.0);
     // fekete a torloszin
   glMatrixMode(GL_PROJECTION);
     // az aktualis matrix mod:  vetites
   glLoadIdentity();
     // a projekcios matrix az egysegmatrix lesz
   gluOrtho2D(0,100,0,100);
     // parhuzamos vetites specifikalasa
}
```


Az `init()` függvény beállítja a törlő színt, a mátrix módot a vetítésre, betölti az egységmátrixot, valamint egy 100×100-as téglalapra fogja elvégezni a vetítést.

Pontrajzoló program II.

```
void display(void) {
  glClear(GL_COLOR_BUFFER_BIT); // kepernyo torlese
  glBegin(GL_POINTS); // pontokat specifikalunk
    glColor3f(1.0f, 0.0f, 0.0f); // piros
    glVertex2i(10, 10); // piros pont
    glColor3f(0.0f, 1.0f, 0.0f); // zold
    glVertex2i(50, 10); // zold pont
    glColor3f(1.0f, 0.0f, 1.0f); // kek
    glVertex2i(30, 80); // kek pont
  glEnd(); // nem lesz tobb pont
  glFlush(); // rajzolj!
}
```


A `display()` függvény fog minden képfrissítésnél lefutni. Először töröljük a képernyőt, majd kirajzoljuk a három pontot. A pontok kirajzolása előtt (ha nem ugyanazzal a színnel rajzolunk, mint korábban meg kell változtatni a színt.
A `glFlush();` függvény fogja kirajzolni a pontokat, amíg nem hívjuk meg, addig nem fog a programunk kirajzolni semmit, hiába adtuk meg a pontokat.

Pontrajzoló program III.

```
void keyboard(unsigned char key, int x, int y) {
   switch(key); // billentyu kod
     case 27:  // ha ESC
       exit(0); // kilepunk
       break;
   }
}
```


A `keyboard()` függvény akkor hívódik meg, ha egy billentyűt lenyomunk. A függvényben le kell kezelni, hogy melyik billentyű hatására mi történjen. Ebben a példában egyszerű a helyzet, ha megnyomjuk az ESC gombot (amelynek a kódja 27), akkor kilépünk a programból.

Pontrajzoló program IV.

```
int main(int argc, char** argv) {
  glutInit(&argc, argv);
  glutInitDisplayMode(SINGLE | GLUT_RGB);
    // az ablak egyszeresen pufferelt, és RGB modu
  glutInitWindowSize(100, 100); // 100x100-as
  glutInitWindowPosition(100, 100); // bal felso sarok
  glutCreateWindow("3point"); // neve 3point
  init(); // inicializalas
  glutDisplayFunc(display); // a kepernyo esemenyek es
  glutKeyboardFunc(keyboard);
    // ...  billentyuzet esemenyek kezelese
  glutMainLoop(); // belepes az esemeny hurokba
  return 0;
}
```


Ez a kódrészlet a főprogram. Minden C nyelvű program belépési pontja a `main()` függvény. (A `main()` függvény két paramétere a program parancssori paramétereinek száma, valamint a paraméterek egy sztring tömbben.)
A `main()` függvény paramétereit továbbadjuk a `glutInit()` függvénynek. Beállítjuk a kirajzolási módot, amely egyszeresen pufferelt és RGB módú, majd megadjuk a program ablakának paramétereit. Ezt követően meghívjuk az `init()` függvényt, majd beállítjuk a képernyő-, valaminta billentyűzet eseményeit kezelő callback függvényeket. (Itt mondjuk meg, hogy a képernyő eseményeit a `display()`, a billentyűzetét pedig a `keyboard()` függvény kezeli le.)
Végül elindítjuk a programciklust, és kilépünk a programból.