

Számítógépes grafika alapjai

Szegedi Tudományegyetem
Informatikai Tanszékcsoport
Képfeldolgozás és Számítógépes Grafika Tanszék
2014-2015. tanév

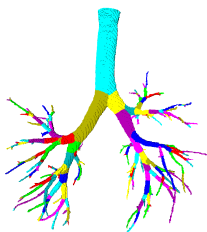
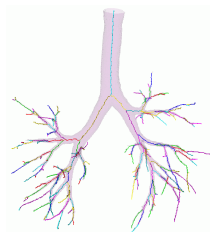
- ▶ Előadó és gyakorlatvezető: Németh Gábor
- ▶ Előadás (nappali tagozaton): heti 2 óra
- ▶ Gyakorlat (nappali tagozaton): heti 1 óra
- ▶ Követelmények:
 - ▶ 1 kötelező program a meghirdetett témakörben
 - ▶ 2 ZH (gép előtt)

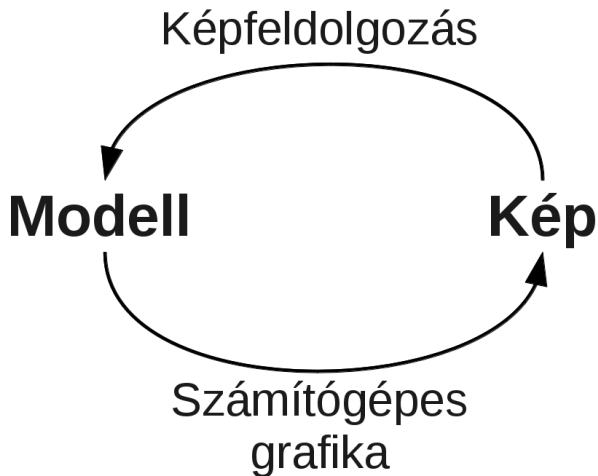
Bevezetés történeti áttekintés

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



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



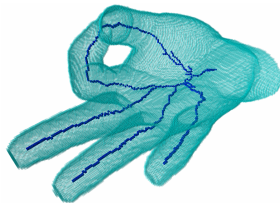


Bevezetés

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

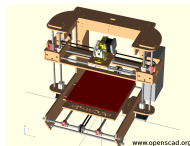
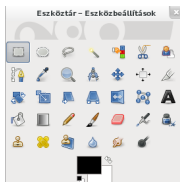
Fejlődés:

fotózás → televízió → számítógépes grafika



<http://www.thenorthernecho.co.uk>

Alkalmazási területek



felhasználói programok üzlet, tudomány

tervezés, CAD

játék, szimuláció



technology.desktopnexus.com

művészet, kereskedelem



git-scm.com

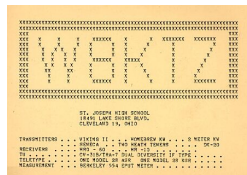
folyamatirányítás



térképészet

Történeti áttekintés

Kezdetben a képek megjelenítése teletype nyomtatókon történt.

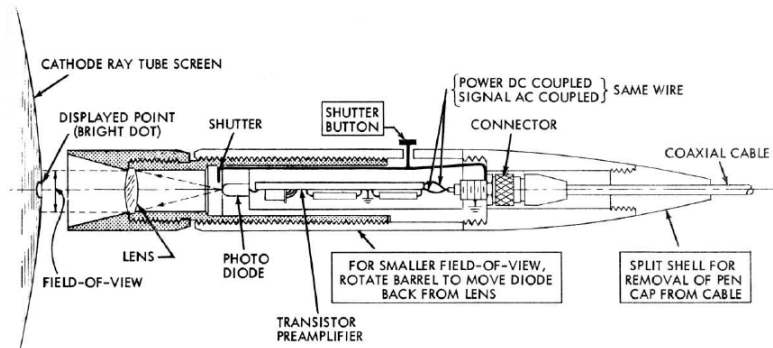


1950:

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



Történeti áttekintés



1963:

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



Történeti áttekintés

1964:

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



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

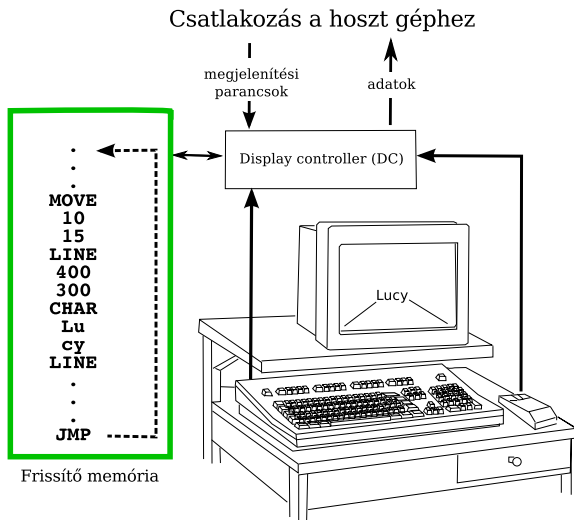
1960-as évek:

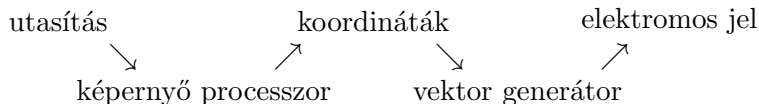
Jellemző output eszköz az ún. **vektor-képernyő** (szakaszokat rajzol ponttól pontig)

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





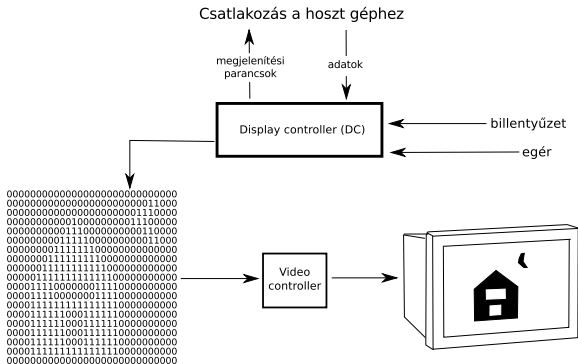
30 Hz-es frissítés (foszforeszkáló képernyő - nem villog annyira)

1960-as évek vége:

DVST (direct-view storage cube) - a látványt közvetlenül tároló
cső: olcsóbb

képernyő $\longleftrightarrow$ kisszámítógép
felszabadul a központi gép

Történeti áttekintés



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

mátrix – raszter sorok – képpontok

Bit-térképek:

$$1024 \cdot 1024 \cdot 1 = 128 \text{ k} \quad \text{bináris kép}$$

Pixel képek:

$$1024 \cdot 1024 \cdot 8 = 256 \text{ szürkeárnyalat vagy szín}$$

$$1024 \cdot 1024 \cdot 24 = 2^{24} \text{ szürkeárnyalat vagy szín}$$

Ma tipikus:

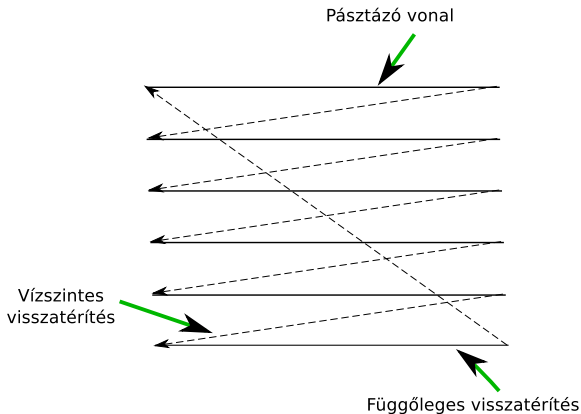
$$1080 \cdot 1024 \cdot 24 \approx 3,75 \text{ MB RAM}$$

A raszteres képernyő előnyei:

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

A raszteres képernyő hátrányai:

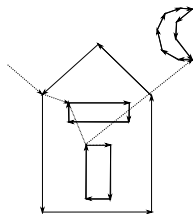
- ▶ A grafikus elemeket (pl. vonal, poligon) át kell konvertálni (RIP - Raster Image Processor)
- ▶ A geometriai transzformációk számításigényesek.



Megjelenítés raszteres képernyőn



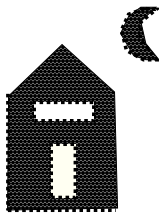
Ideális vonalrajz



Vektoros kép



Raszteres kép



Raszteres kép terület kitöltéssel

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

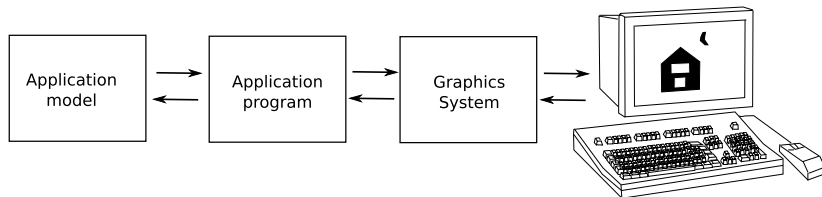
Történeti áttekintés (szoftverek)

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)



Interaktivitás

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

Felhasználói modell adatbázis:

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

Az interaktivitás 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
- ▶ szürkeárnyaltos
- ▶ 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:

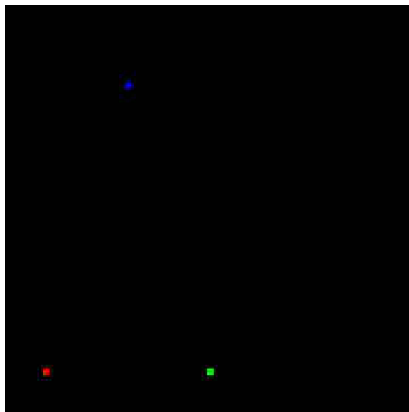
- ▶ Végtermék
- ▶ Közbülső termék

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)



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 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);
GLclampf - float

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 színbeállítás 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

Csúcsopontok megadása

Csúcspont(ok) (vertexek) megadása:

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

Például:

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


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

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.

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);
```

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)
```

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

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

`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.

`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)`;

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
}
```

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!  
}
```

Pontrajzoló program III.

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

Pontrajzoló program IV.

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