Bevezetés a CGI-be

1. Történelem

1.1 Úttörők

- Euklidész (ie. 300-250)
  - A számítógépes grafika geometriai hátterének a megteremtője
- Bresenham (60’ évek)
  - Első vonalrajzolás “raster” készüléken, később kibővítette kör és anti-aliased vonalakra
- Gouraud és Phong (70’ évek)
  - Felfedezték a “rendering(shading)” eljárásokat
- Jim Blinn (80’ évek)
  - Texture mapping

1.2 Hardware

- Oszcilloszkóp: „Tennis for two”
- Televízió, RGB monitorok, LCD handheld, Atari, Famicom(8bit)
- RenderMan: Toy Story
- Voodoo grafikus akcelerátor
- Fixed function pipeline
- GeForce1 első Transform&Lighting-al rendelkező akcelerátor
- nVidia – CG nyelv
- Programozható vertex-shadereket támogató DirectX verziók (8+)
- Programozható pixel shaderek megjelenése

Bevezetés a CGI-be

2. Képfeldolgozási alapok

2. Képfeldolgozási alapok

- 2.1 A “SetPixel” metafüggvény
- 2.2 Flipchains
- 2.3 Színezés

A raszterizáció

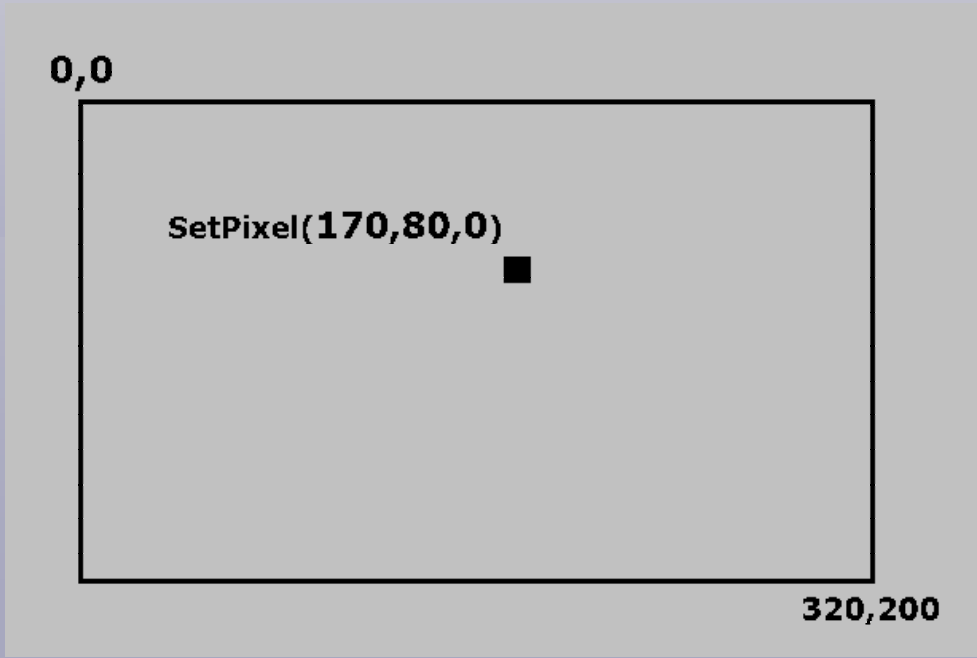
Az a folyamat amely során a rasztertárat adattal töltjük fel

- Egy raszter készülék a rasztertár adatait olvassa ki
- Rasztertár – Lineáris memóriarész ami képpontokat tartalmaz (frameBuffer)
- Képpontok(pixel) – színkódot tartalmaznak, helyük a memóriában határozza meg a koordinátákat a raszter megjelenítőn

A SetPixel algoritmus

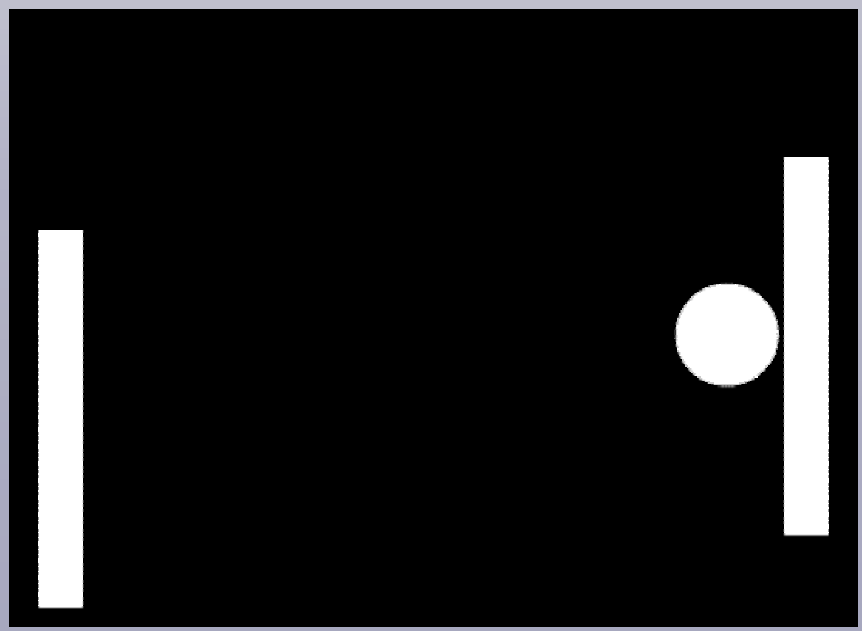
SetPixel(x, y, szín)

frameBuffer[x + y * szélesség] = szín



Backbuffer

- Amikor közvetlenül a Videómemóriaba írunk, fellep a villogás hiba (flickering)
- Backbuffer: a rendszer memóriájába „rajzoljuk” a képet, majd az egész tömb tartalmát bepakoljuk a videómemóriába - a kép így csak akkor frissül amikor készen vannak a rajzolási műveletek
- Flipchains: Általában több ilyen backbuffert szoktunk használni



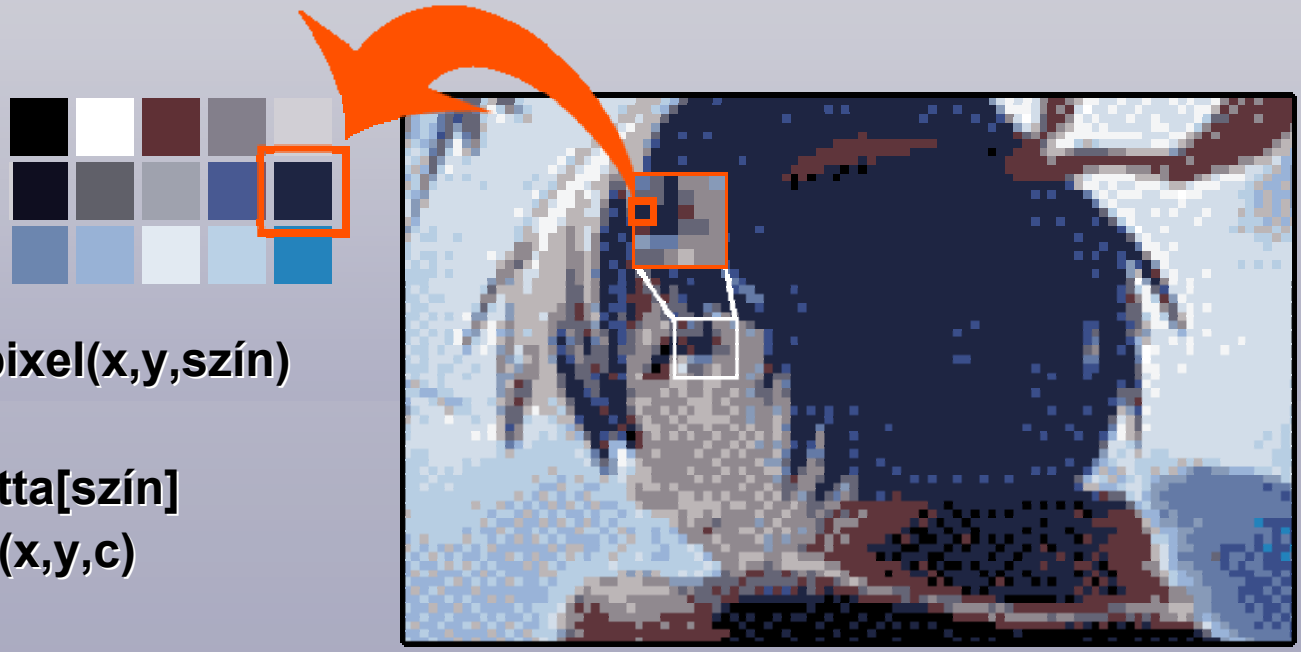
Színösszetétel

Red Green Blue (RGB)

- Egy színkód 3 darab 8 bites komponensből áll
  - pl: RGB(255,0,0) a “legpirosabb” szint állítja elő”
- maximálisan: $256^3 = 16.777.216$ darab szín
- RGB monitor esetén maximálisan a 24 bites színkódolás, ezen felül kizárólag csak az áttetszőségre használandó
- Egyéb színformátumok: CMYK, BGR

Paletták

- Minél nagyobbak a színkomponensek, annál lassúbb a kép összetevése
- Hardware korlátok
- Paletta: előre definiált, indexelt színkód tábla



```
set_palette_pixel(x,y,szín)
{
        c = paletta[szín]
        setpixel(x,y,c)
}
```

- Lehetőségünk van különféle un. „paletta effektusokra”

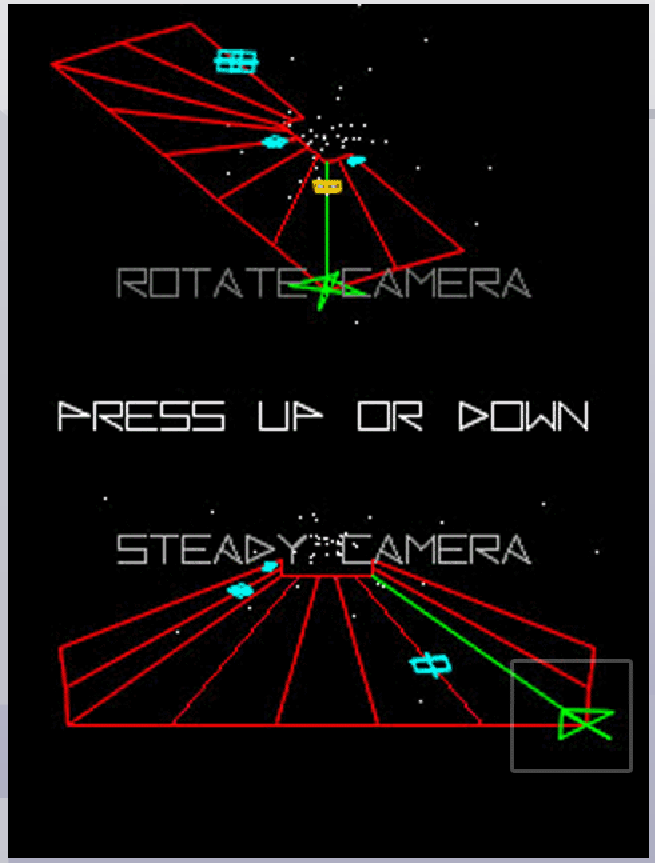
Bevezetés a CGI-be

3. Vektorgrafika

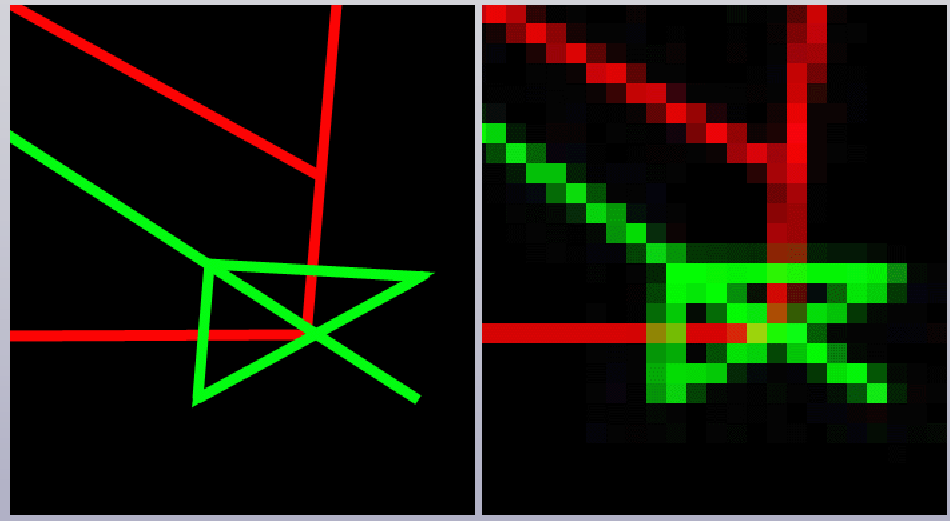
3. Vektorgrafika

- 2D vektorgrafika
- 3D vektorgrafika
- Transzformációk

Vektorgrafika



Közelítés



Vector Sprite

- Rezolúciófüggetlen
- Memória-barát – algoritmus „rajzolja” ki a megadott alakzatot, nem bittérkép kódolja
- Geometriai vetítések raszterizálás

Transzformációk

- Minden vektor átmegy a transzformációs lépéseken
- 1. WORLD
  - Minden hozzárendelt vektor-komponenst módosít
  - Módosítók: Rotate, Scale, Translate
- 2. VIEW
  - Beállítunk egy nézőpontot és egy nézet irányt
- 3. PROJECTION
  - Nézetmező (FOV), nézet-szög

Bevezetés a CGI-be

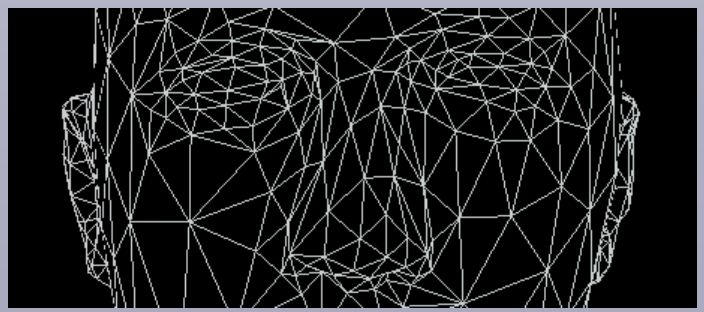
4. Háromszögek, textúrák

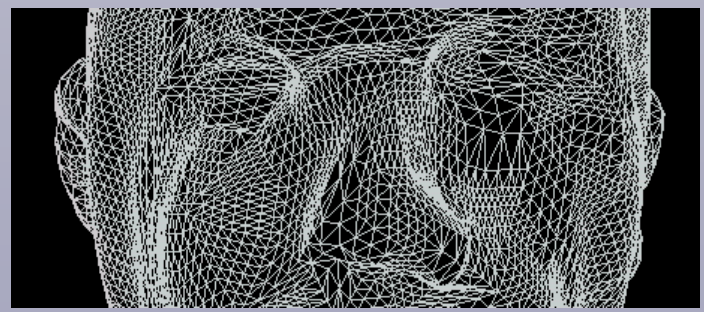
4. Háromszögek, textúrák

- Háromszögek a CGI-ben
- Textúrálás
- Meshek

Háromszögek a CGI-ben

- Vertice / Vertex – olyan adatstruktúra, ami egy vektor koordinátái mellett egyéb információt is tartalmazhat (például: pozíció, textúrakoordináták, normálvektor)
- 3 vertice meghatároz egy háromszöget
- a 3D grafika alapja a háromszög
- a legkomplexebb modellek is háromszögekből állnak





Texturálás

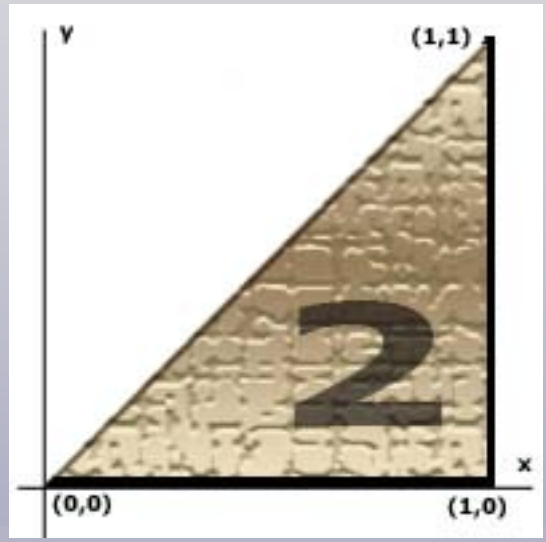
Nagy mértékű realizmust nyújt a képnek, minimális erőforrás befektetéssel.

- képet rajzolunk a háromszög felszínére
- Nem módosul a poligonok / vertex-ek száma
- Mintákat lehet készíteni
- Általában hibrid módszereket alkalmazunk
- Természetből választott minta vs. procedurális minta

Textúra koordináták

- Egy háromszög textúra koordinátákkal rendelkezhet

Pl: V1 (0,0,0,) texturecoord (0,0)
V2 (1,0,0,) texturecoord (0.5,0)
V3 (1,1,0,) texturecoord (0.5, 0.5)



Képpont egy vertex esetén:
TU * képszélesség
TV * képmagasság

- A grafikus hardware (vagy shader program) feladata a háromszög effektív textúrálása (általában raszterizáláskor történik)

Meshek

- Szükség van egy csoportosításra, az „egy tárgyhoz tartozó” háromszögek esetén

- Helyezés, forgatás és skálázást egységesen kell végrehajtani

- Megjelenik a „pivot point”, azaz mesh „középpontja” (például elhelyezni a mesht úgy lehet, hogy a pivot point koordinátáit toljuk el, a többi automatikusan adaptálódik)

- Animáció, fényezés, anyag-definíció (material) és egyéb effektusokra lesz szükség

Bevezetés a CGI-be

5. Akcelerátorok

5. Grafikus Akcelerátorok

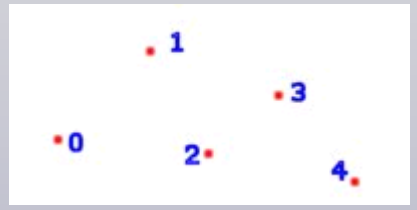
- Applikáció
- Illesztő API-k

- Egyszerű példa: “the Spinning Cube”

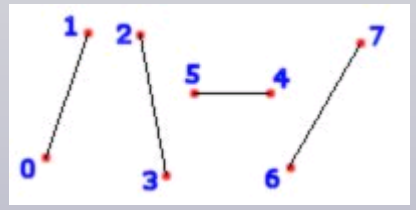
Valós idejű rajzolás

- **INTERAKTIVITÁS**
- Shader – egység ami meghatározza a háromszögeket és textúrázza őket
- Rendering – az a folyamat amely során a Shader megkapja a kép összetételéhez szükséges adatokat és megtörténik a Shading illetve a Raszterizálás

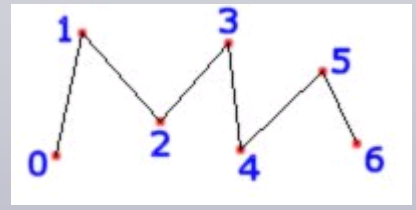
Primitív típusok



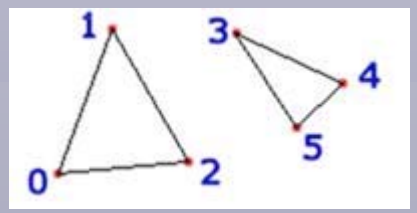
Pont Lista



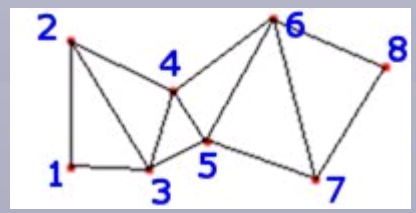
Vonal Lista



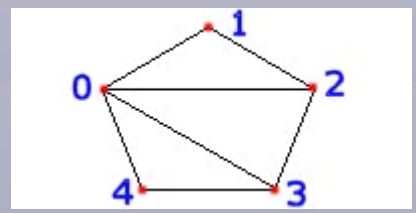
Vonal sáv



Háromszög lista



Háromszög sáv



Háromszög „fan”

the Spinning Cube

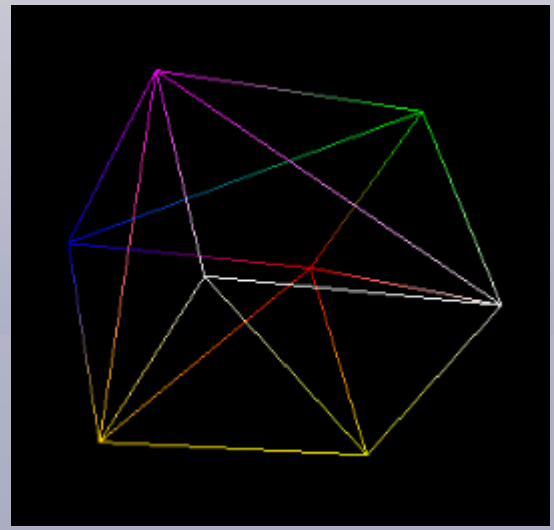
Feladat: Létrehozni egy forgó kockát a grafikus akcelerátort használva.

Lépések:
-Inicializálni az API-t és az akcelerátort

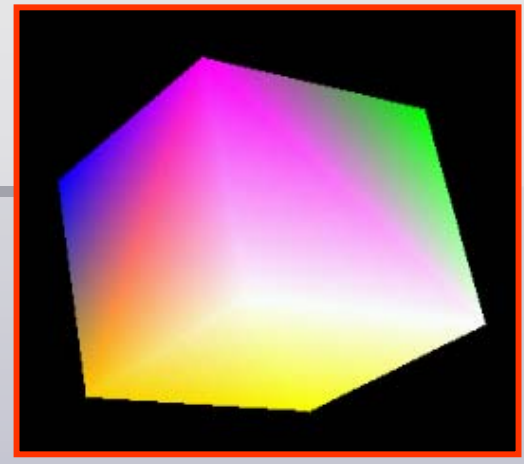
-Feltölteni a VertexBuffert
-Feltölteni az IndexBuffert
-Beállítani a mátrixokat (world, view, projection)

-Elkészíteni a „render” loop-ot ami folytonosan frissíti a képet
- -Beállítani a tanszformációkat
- -Beállítani a StreamSource-ot
- -Rajzolási lépés
- -Page-Flip

-Felszabadítani a használt eszközöket



Vertex, Index buffer



Probléma:

-egy kocka esetén, tarolni kéne 36 darab vertice-t

-egy Kockának csak 8 darab csomópontja van

Megoldás:

-Tároljuk a 8 csomópontot egy tömbbe

-A háromszögek meghatározásához, csak indexeket használjunk

pl: 1 vertice mérete (x float, y float, z float, tx float, ty float)

= 20Byte

1 index mérete (word) = 2Byte

-Kocka definiálása indexbuffer nélkül: 36x20Byte = 720Byte

-Kocka definiálása indexbufferral: 8x20Byte + 36x2 Byte = 232Byte

A Memóriahasználat az eredeti 32%-ára csökkent