Dr. Mileff Péter

GRAFIKA PROGRAMOZÁSA

BEVEZETÉS

Miskolci Egyetem
Általános Informatikai Tanszék

Bemutatkozás és követelmények

- **Dr. Mileff Péter**
  - Helyileg: Informatikai Intézet
    - 110. szoba
  - Konzultációs idő: Szerda 10-12
    - mileff@iit.uni-miskolc.hu
- **Követelmények:**
  - Az órák ¾-én kötelező a megjelenés.
  - Zárthelyi dolgozat nincs
  - Féléves feladat van. Csoportokban végezhető.
  - Zárás:
    - Féléves feladatra kapott jegy

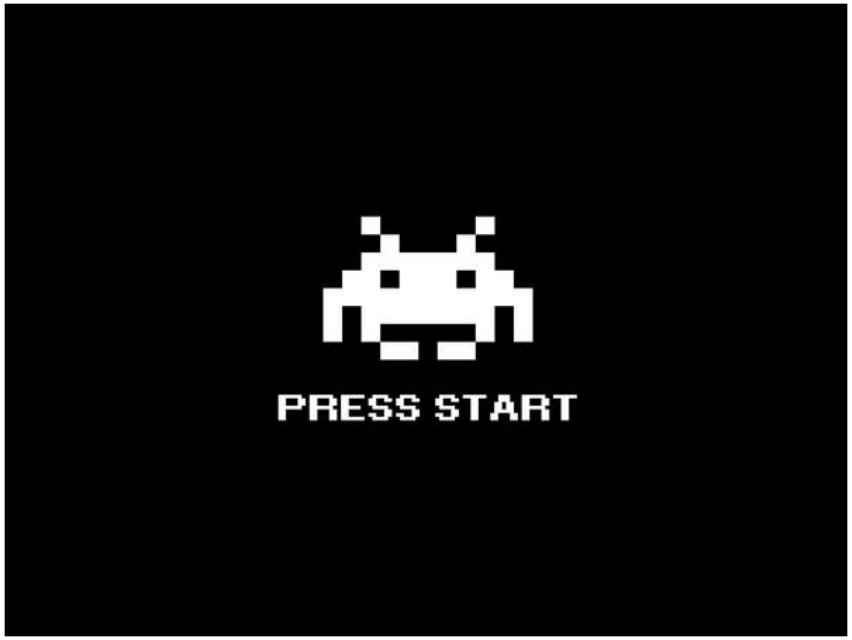
Jegyzet: http://www.iit.uni-miskolc.hu/~mileff/grafika

Féléves feladat

- Csoportokban végezhető: maximum 3 ember
- Követelmények:
  - Valamilyen technológiai jellegű bemutató program:
    - Demo, játék, stb
    - Pl: benchmark program mobil eszközökre
  - Lehetőleg OpenGL API-t használjon
  - Operációs rendszer: nincs megkötés
  - Nyelv: nincs megkötés
  - Játék készítő programok kerülendők:
    - Pl. Game Maker
- **Határidő**: félév vége

Miről lesz szó?

- A tárgy célja: megismertesse a hallgatókat a játék fejlesztés és a grafika világával
  - Módszerek, megközelítések, megoldások
- Főbb témák:
  - Számítógépes grafika alapjai, fejlődése
    - Grafikus és játékmotorok
    - GPU technológiai áttekintés
  - Gyakorlati 2D grafika
    - Renderelés alapjai, nehézségei
    - Objektumok mozgatása, animáció
    - Befoglaló doboz, ütközésvizsgálat, vágások
    - Tilemap, bitkép szövegek
- Gyakorlati 3D grafika
  - Programozható csővezeték, árnyalók alkalmazásai
  - Fények és árnyékok
  - Effektek: bump, normal mapping, ambient occlusion, stb



„A GPU erős számolási csodaeszköz. A GPU aritmetikai teljesítménye, egy magas szinten specializált architektúra, több éves fejlődésének eredménye. Az egyre növekvő sebesség és rugalmasság miatt sok fejlesztő leleményességének eredményeként számos olyan alkalmazás létezik, amely a GPU-t nem grafikai feladatokra használja. De sok olyan alkalmazás is létezik, amely sosem lesz képes a GPU lehetőségeit kihasználni.

A szövegszerkesztés, például egy tipikusan olyan „klikkelős" alkalmazás, amely sosem fogja tudni kihasználni a GPU párhuzamosítási lehetőségeit."

[John D. Owens 2005]

SZÁMÍTÓGÉPES GRAFIKA ALAPJAI…

Bevezetés

- A számítógépes grafika életünk szerves részét képezi.
  - Gyakran észrevétlenül, de szinte mindenhol jelen van a mai világban.
    - Pl.: játékok, filmek, reklámok, plakátok, stb.
- A terület sok éves fejlődésen ment keresztül az utóbbi néhány évtizedben
  - Több platform is megjelent: C64, ZX Spectrum, Plus 4, Atari, Amiga, Nintendo, SNES, stb
  - Fő indítója a személyi számítógépek (PC) megjelenése.
- A PC-k megjelenése és otthonokba való eljutása drasztikus növekedésnek indította **videojáték ipar**t.
  - Lehetőséget a játékok otthon való használatára, valamint a grafika programozására egyaránt.

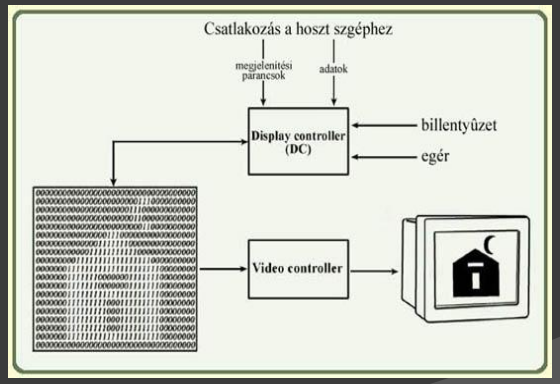
Bevezetés

- Ma már kijelenthetjük: a számítógépes vizualizáció fejlődését a média és videojáték ipar irányítja.
  - Konzol, PC és ma már mobil platformok is
- **Jellemzőik:**
  - Folyamatos intenzív fejlődés
  - A képi világgal szemben támasztott egyre magasabb igények
  - Új grafikus algoritmusok kidolgozása
  - A valóság minél pontosabb közelítése
- Fontos állomása a grafikus célprocesszorok megjelenése volt
  - számtalan új lehetőséget nyitott meg a fejlesztők előtt
    - nem csak a megjelenítés gyorsítása, hanem általános célú számítások (Tendálás a CPU felé)

A számítógépes grafika feladata

- A memóriában tárolt adatok, objektumok, primitívek átalakítása és leképzése a két dimenziós síkra, a képernyőre.
  - Erre különböző algoritmusok alakultak ki.
- Összefoglaló neve: **Raszterizáció**
- A megjelenítés legkisebb egysége: **pixel**.
  - Egy önállóan megjeleníthető pont raszteres grafikus eszközökön (képernyő, nyomtató stb.).
  - Színe a színtér által meghatározott.
    - Pl. RGB, RGBA, HSL, HSV, CMYK
- A teljes kép pixelek halmaza - mennyiség felbontás függő

Raszter grafika



Raszterizáció irányai

- A raszterizációs megoldások **két irány** köré csoportosulnak:

1. A valóság minél pontosabb modellezése:

- Cél: a magasabb képi minőség elérése
- Főkét tervező és modellező programok biztosítanak.
- A megjelenítés ekkor nem valós időben történik
  - a magas képi minőségből fakadó számítási időigény miatt
- Képviselők: sugárkövetés, Photon Mapping, Radiosity, stb

2. Gyors, valós idejű modellezés:

- Cél: valós időben végzett raszterizáció legalább 60 FPS-el
- Legfontosabb képviselői: számítógépes játékok, demók és egyéb kompozíciók

KORAI SZÁMÍTÓGÉPES GRAFIKA FEJLŐDÉSE…

Korai megjelenítés

- A korai számítógépekben nem létezett GPU.
- A grafikai számításokat a központi egység (CPU) végezte el.
- A CPU-k az általános célú programok számára készülnek
  - mind a mai napig nem tartalmaznak speciális funkciókat ellátó hardverelemeket
  - Utasításkészleteket igen (pl. 3DNow - AMD)
- Korán felismerték a grafika gyorsíthatóságát:
- Például:
  - C64 – hardveres sprite-ok és scroll,
  - AmigaAGA chipset (A1200 and A4000)

Szoftveres raszterizáció

- A számítógépes grafika korai vizualizációs programozási modellje.
  - Szoftveres raszterizáció - software rendering/rastering
  - A számítógépes grafika első lépcsője
  - Kb 2002-ig tartott.
- **A megközelítés modellje:**
  - Minden számítási feladatot a központi egység végzett.
    - Gyakorlatilag nem is volt más egység, amely elvégezhette volna, így maradt a CPU.
  - A megjelenítendő képet egy a CPU által futtatott szoftveres program fogja előállítani.
  - Az megjelenítési forma elnevezése (szoftveres) is erre utal

Szoftveres raszterizáció

- Az alakzatokat felépítő geometriai primitívek:
  - 2D esetén téglalap, 3D esetén háromszög
  - a központi memóriában helyezkednek el tömbök, struktúrák és egyéb elemek formában.
- A központi egység (CPU) ezeken végzi el a kérdéses műveleteket
  - színezés, textúra leképzés, színcsatornák állítása, forgatás, nyújtás, eltolás, stb),
  - A képet egy speciális tömbben a **Framebuffer**-ben tárolja
  - Végül küldi ki a elkészített képet a video vezérlőnek.

SZOFTVERES RASZTERIZÁCIÓ ELŐNYEI…

Szoftveres raszterizáció előnyei

- A szoftveres renderelés virágzó időszaka a DOS alatt.
- **Fejlesztések fő jellemző:**
  - Alacsony felbontás (320x240, 640x480, 800x600) – VESA mód
  - 256 szín, Watcom C fordító, DOS/4GW – 32 bites DOS extender
- Hatékony, hardver közeli megjelenítés.
- **Oka:**
  - A DOS operációs rendszer egy egyfelhasználós operációs rendszer volt,
    - kizárólag egy darab taszk futhatott egyszerre.
  - A video memóriát közvetlenül lehetett címezni a felhasználói programból (pl. 0xA0000 címen)
    - A címzés kapcsán az adat (pixel) azonnal megjelent a képernyőn.

Szoftveres raszterizáció előnyei

- **A programozónak teljes teljhatalma volt:**
  - Minden képernyőpontot (pixel) egyedileg tudott kezelni
  - és vezérelni a kirajzolási folyamatot.
- **Rendkívüli rugalmasságot biztosított:**
  - összehasonlítva a mai GPU támogatott rendszerekkel
  - Nem volt szükség a grafikus processzorok programozási nyelvének elsajátítására,
  - a programkód tiszta, logikus és „egyszerű” volt.
  - A grafikus csővezeték egésze programozható volt a programozó által.
  - Mivel a video memória kezdőcíme egységes volt a VESA nemzetközi szabványnak köszönhetően így a megoldás platform független is volt egyben.

Szoftveres raszterizáció előnyei

- A mai videokártyák esetében ez nem így van.
- A kártyák funkcionalitása és programozhatósága korlátozott.
  - Bár folyamatosan fejlődik
- Minden kártya csak egy megadott verziószámú árnyaló (shader) modellt támogat.
- A megfelelő GPU-val nem rendelkezve gyakorlatilag az adott szoftver sokszor nem futtatható.

SZOFTVERES RASZTERIZÁCIÓ HÁTRÁNYAI…

Szoftveres raszterizáció hátrányai

- A grafikában fontos cél a gyors valós idejű vizualizáció elérésre.
  - Ehhez nagy adathalmazt kell mozgatni és kezelni.
- A szoftveres megoldás legfontosabb nehézsége ebben van.
- **Minden adat a központi memóriában van tárolva:**
  - Bármely módosítás esetén mindig a központi memóriához kell fordulnia a CPU-nak.
  - Ezeket a kéréseket korlátozza a buszra kiadható legnagyobb bitmennyiség értéke,
  - valamint az adott memóriatípus elérési ideje

Szoftveres raszterizáció hátrányai

- **Ügyelni kell az adatok elhelyezésére:**
  - A memória legkevésbé legyen szegmentált!
  - A szegmentáltan adatokon végzett gyakori módosítások nagy lassulásokat eredményeznek.
  - **Oka:** Cache hibák magas száma
    - A nem folytonos és rendezett adatokon a cache sem segít
    - Folyamatosan cserélődik a tartalma
- A kódnak erősen optimalizáltnak és alacsony szintűnek kellett lennie
  - Kritikus részek: C, ASM

Szoftveres raszterizáció hátrányai

- **Nagy adathalmaz mozgatása a központi memória és a video memória között:**
  - Korlátja a busz sávszélesség
  - A folyamatos megjelenítéshez mindezt 50-60-szor kell megtenni.
    - Jelentős erőforrás
- **Kezdetben kezelhető volt:**
  - Videokártyák, monitorok alacsony felbontásúak voltak
  - Tipikus akkori játék: 640x480, 800x600, esetleg 1024x768
  - A mozgatott adathalmaz ekkor nem olyan nagy:
    - Pl. 640x480 képpont felbontású képpek, 8 bites színmélységgel számolva 640*480*1byte=30720byte=300 Kb 1 képkocka
    - 1024x768 képpont esetén 768 Kb 1 képkocka

Szoftveres raszterizáció eredményei

- Az optimalizáció révén nagyszerű látványvilággal dolgozó szoftverek születtek:
  - A CPU “rohamos” fejlődése jó alapot adott
    - MMX, 3DNOW utasításkészletek
  - C nyelv egyetemes elterjedése és az Assembly nyelv kombinációja ütős páros volt
    - a váz C/C++ -ban készült
    - a kritikus részem assembly blokkokban

Kiemelkedő eredmények

Doom - 1993

- **2.5 D grafika**
- **BSP térfelosztás**
- **Raycasting, fények**

Kiemelkedő eredmények

Quake I - 1996

- Valódi 3D grafika
- Poligon modellek
- BSP térfelosztás
- Lightmaps, valós idejű fények
- MMX optimalizált - Michael Abrash

Kiemelkedő eredmények

Unreal Engine 1 – 1996 - 1998

- Valódi 3D grafika
- Poligon modellek
- BSP térfelosztás
- Lightmaps, valós idejű fények
- Köd, MMX optimalizált

SZOFTVERES RASZTERIZÁCIÓ JÖVŐJE…

Szoftveres raszterizáció jövője

- Egyeduralkodó volt az első GPU-k megjelenéséig (1996)
- Egy ideig párhuzamosan futott a két megjelenítés
  - a gyakorlatban, mára a szoftveres megjelenítés szinte teljesen eltűnt
- **Okai:**
  - 1) a buszsebesség korlátja
  - 2) memóriák sebessége: a központi memóriák egy generációval lemaradtak a GPU memóriákhoz képest
    - GPU - DDR5, CPU - DDR4 (bevezetés alatt)
  - A központi egység sebessége nagyon erősen megnövekedett
  - De nem tudta tartani a versenyt az újonnan bevezetett GPU architektúrával

Quake 1 - GPU vs CPU

A GPU-k ugrásszerűen megnövelték a grafikai minőséget

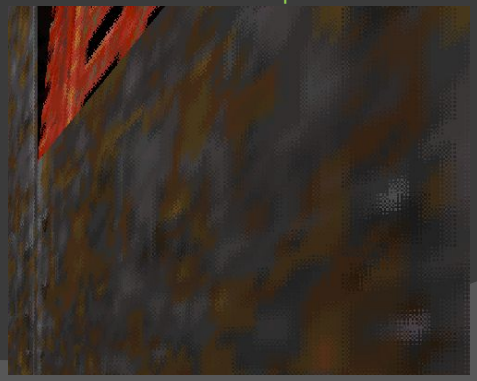
Szoftveres raszterizáció jövője

- A szoftveres megjelenítés jellegzetessége a “pixelesség” volt
  - A két dimenziós játékokban nem igazán volt megfigyelhető
    - ott nem nagyon alkalmaztak textúra nyújtásokat és így szűréseket sem a játékmenet során.
  - Három dimenzióban, főként FPS játékokban:
    - a falhoz közel állva megfigyelhettük annak pixelességét
    - Oka: az akkori CPU teljesítmények mellett nem alkalmaztak textúra szűréseket (pl. bilinear filtering)
      - túl magas volt az erőforrásigénye
      - már az első GPU-k ezt hardveresen tudták!

Szoftveres raszterizáció jövője

- Modernebb CPU-k új lehetőséget nyitottak: MMX, SSE, SSE2
- Pl. Unreal tournament szoftveres textúra szűrője:

ordered texture coordinate space dither



Szoftveres raszterizáció jövője

- **Napjainkban:**
  - ma már szinte minden eszköz kap egy GPU-t
  - GPS, mobiltelefonok, tabletek, kézi konzolok, stb,
  - Programozásuk viszonylag egységes: **OpenGL, OpenGL ES**
- **Magától értendő, hogy a GPU alapú szoftverek készülnek**

Szoftveres raszterizáció jövője

- A megjelenítési technika nem tűnt el:
  - Szerepe a következő években ismét előtérbe fog kerülni:
    - az érkező sokmagos processzorok miatt
    - az új utasításkészletek miatt: SSE4.2, AVX, AVX2, stb
- 2014-ben megjelent új DDR4 típusú központi memória modulok
- A fejlesztők újragondolják a grafikus alkalmazásaik felépítését, logikai modelljét:
  - AAA játék számára a multithread modell már elengedhetetlen
  - **Feladatok megosztása a CPU és GPU között:**
    - Szoftveres occlusion culling (pl. KillZone 3, Battlefield 3)

Szoftveres raszterizáció jövője

- Napjainkban három DirectX 9 kompatibilis sz. renderer:

  1. Pixomatic renderer – RadGameTools
  Non free licence
  2. **Swiftshader** – TransGaming
  Van kipróbálható verzió

- Mindkét fejlesztés jól optimalizált: kihasználja a modern processzorok utasításkészletét (pl. SSE, AVX,MMX,3DNow)

- 3. **DirectX WARP** – Microsoft szoftveres renderer

Utolsó nagy eredmény

Unreal Tournament 2004 – Pixomatic renderer

GAME OVER