Peter Mileff PhD

Programming of Graphics

Introduction

University of Miskolc
Department of Information Technology

Task of the semester

- **Implement a graphics based demo application**
  - Any application which focuses mainly to 2D or 3D graphics
  - **Preferred:** simple game or technological demo
    - e.g. collision test, screensaver, animation, etc
  - Use the OpenGL API if possible
  - **Any platform:**
    - Android, iOS, PC, Java, C++, etc
- Deadline: end of the semester

The main topics

- Fundamentals and evolution of computer graphics
- Overview of GPU technologies
- Game and graphics engines
- Practical 2D visualisation
  - Basis and difficulty of rendering - software rendering
  - Moving objects, animation
  - Collision detection
  - Tilemap, bitmap fonts
- Practical 3D graphics
  - Object representation, structure of a model, rasterization algorithms
  - Programmable Pipeline, applying shaders
  - Lights and shadows
  - Effects: bump mapping, normal mapping, ambient occlusion, etc
  - Billboarding, terrain rendering, particle effect etc
- Raytracing, Voxel based visualisation

PRESS START

Fundamentals of computer graphics…

Introduction

- Computer graphics forms an integral part of our lives
  - Often unnoticed, but almost everywhere in the world today
    - E.g: Games, Films, Advertisement, Posters, etc
- The area has evolved over many years in the past few decades
  - Multiple platforms appeared: C64, ZX Spectrum, Plus 4, Atari, Amiga, Nintendo, SNES, etc
  - The appearance of the PCs was a big step
- **The video game industry started to grow dramatically** because of PCs
  - Possibility to play game at home
  - Possibility to programming (at high level) graphical applications

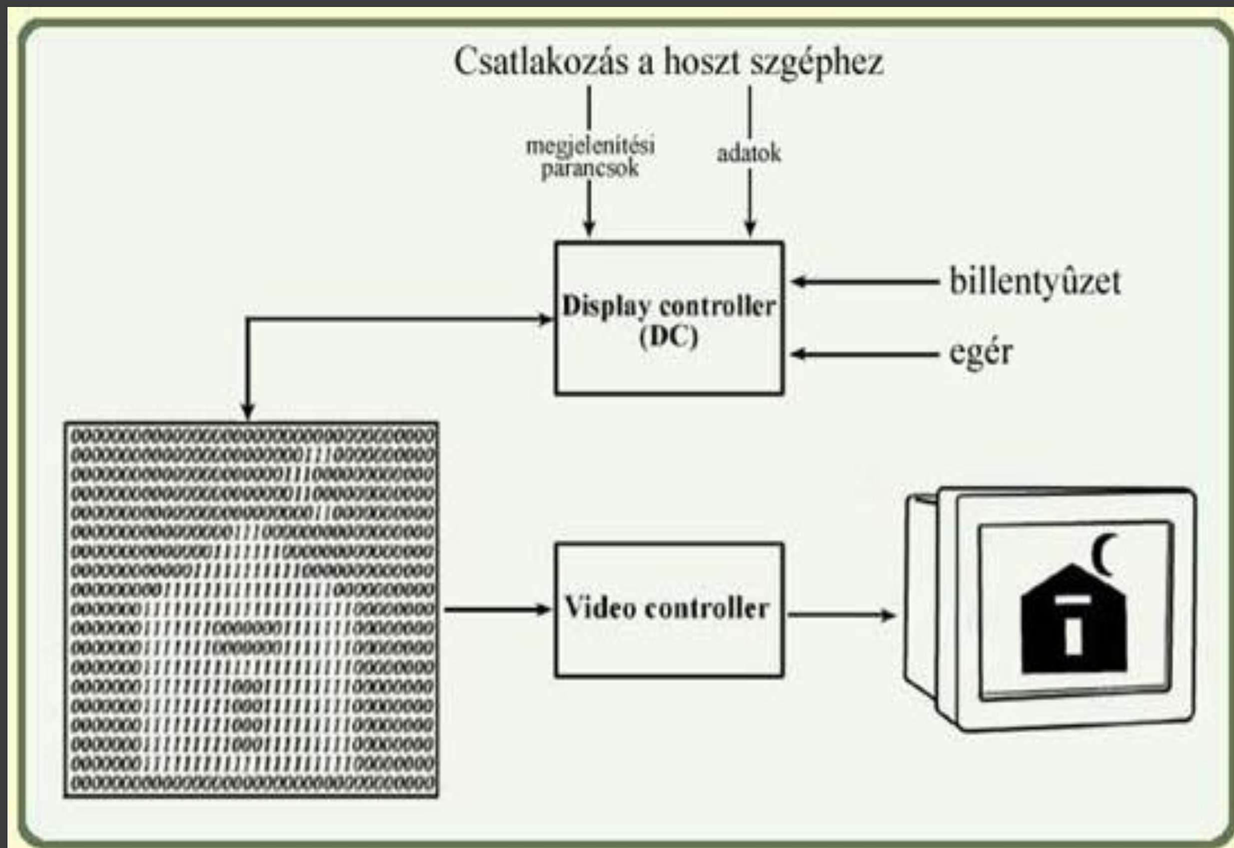
Introduction

- **Today we can say:** the media and video game industry controls the development of computer visualization.
  - Consoles, PC and smartphones today
- **The characteristics of the area:**
  - Continuous, intensive development
    - New algorithms, models, approaches
  - Increasingly higher demands against the visual quality
  - More realistic, physically based rendering
- An important milestone was the appearance of graphic processors
  - opened numerous new opportunities to developers
    - not only accelerate the rendering, but applying general purpose calculations (Like a CPU)

The task of computer graphics

- Mapping and transforming objects, primitives located in main/GPU memory to the two-dimensional plane of the screen
  - Variety of algorithms have emerged for this
- Global name: **Rasterisation**
- The smallest unit of display: **pixel**
  - An independent displayable point of raster graphics devices (Monitor screen, printer, etc.)
  - Its color is specified by the color space
    - E.g.: RGB, RGBA, HSL, HSV, CMYK
- The complete screen is a set of pixels
  - 2D array
  - its quantity is resolution dependent

Raster graphics



Directions of rasterization

- The rasterization solutions can be grouped into **two directions**:

  1. Modeling the reality more precisely:
  - Objective: achieve high and realistic image quality
  - Mainly design and modeling programs
  - The visualisation is not real time (although there are some new approaches doing it quasi real time)
    - Due to the high computational time of realistic rendering
  - Models: Ray Tracing, Photon Mapping, Radiosity, etc

  2. Fast, real time visualisation:
  - Objective: perform rendering in real time at ~60 FPS
  - Main directions: computer games, demos and other compositions

THE EVOLUTION OF EARLY COMPUTER GRAPHICS…

The former visualisation

- The old computers did not have GPU
- The graphical calculations was carried out by the central processing unit (CPU)
- The CPUs are developed for general-purpose programs/calculations
  - did not contain any specific hardware components
  - but can contain specific instruction sets (e.g. 3DNow - AMD, SSE family, Altivec, NEON, etc)
- It has been recognized early, that graphics can be accelerated
- For example:
  - C64 – hardware based sprites and scroll
  - Amiga AGA chipset (A1200 and A4000)

Software rendering

- The early programming model for computer graphics
  - The first stage of Computer Visualisation
  - The model was used about until 2002
    - in parallel with the GPU for a while
- **The model of the approach:**
  - Every computing task was performed by the CPU
    - There was no other unit at time
  - The raster image is calculated by a software program
    - running at the CPU

Software rendering

- **The geometric primitive of the shapes:**
  - they are located in the main memory in form of arrays, structures and other representations
  - Generally:
    - In case of 2D: rectangle (2D array) or two triangle
    - In case of 3D: triangle or voxel
- The CPU performs the actual operations on these primitives
  - coloring, texture mapping, adjusting color channels, rotate, scale, translate, etc,
  - The final image is stored in a special 2D array: **Frame Buffer**
  - At the end of rasterization, this array is sent to the video controller

Benefits of software rendering…

Characteristics of early software rendering

- The prosperous period of this model was during the DOS
- **The main characteristics of the developments:**
  - Low resolution (320x200, 320x240, 640x480, 800x600) – VESA mode
  - 256 colors, Watcom C compiler, DOS/4GW – 32 bit DOS extender
    - DOS was a 16 bit OS, extender was needed to use more memory
- Effective and hardware close visualization:
  - DOS was a single user, single task operating system
    - Can run only one program or application at a time
  - Video memory addressing can be performed directly from the user program (e.g.: video memory starting address 0xA0000)
    - With addressing pixels immediately appeared on the screen

Benefits of software rendering

- **The developer has full control over everything:**
  - Programming every screen pixel singly
  - Control the whole rendering process
- **Offered an extraordinary flexibility:**
  - Compared to today's GPU support systems
  - There was no need to learn the language of programming graphics processors
  - The source code was clean, logical and “simple”
  - All the parts of the graphics pipeline was programmable by the developer
  - The solution was almost platform independent, because of VESA

Benefits of software rendering

- **Today's video card is limited:**
  - in functionality and programmability
    - Although they are continuously evolving
- Every card supports only a specific version of:
  - a shader model
  - a graphics API (OpenGL, DirectX)
- Without the proper GPU, generally the software cannot be run

DRAWBACKS OF SOFTWARE RENDERING…

Drawbacks of software rendering

- The major goal of the graphics is to reach high quality, fast real-time visualization
  - To achieve this large data sets need to handle and move
- This is the main difficulty of the software model
- **All data is stored in main memory:**
  - In case of any change in data, the cpu always needs to communicate to this memory
  - These requests are limited by the BUS speed
    - and the access time of the actual memory type

Drawbacks of software rendering

- **The programmer should care of data structures:**
  - It is not good when memory is segmented!
  - Frequent changes on a segmented memory area kills the performance
  - **The reason is the huge number of cache misses**!
    - If data area are not ordered and continuous, the cache cannot help
    - The content of the cache is always refreshed to gain the required data
- The code should be highly optimized and usually low-level
  - The critical parts are typically in C and ASM

Drawbacks of software rendering

- **Moving large amounts of data between main memory and video memory:**
  - Limited by the BUS speed
  - Continuous visualization requires 50-60 refresh rate per min.
    - This is a significant performance requirement
- **In the beginning the problem was manageable:**
  - Video cards, monitors supported only low resolutions
  - Typical game at that time: 640x480, 800x600 or maybe 1024x768
  - <u>The moved data set is not so large:</u>
    - E.g.: 640x480 screen resolution, 8 bit color depth results: 640*480*1byte=30720byte=300Kb / 1 frame
    - In case of 1024x768 one frame is 768 Kb

Milestones of software rendering

- Due to proper optimizations and efficient algorithms, many software (with great visual) were developed
  - The rapid development of the CPU given a good basis
  - The universal spread of the C language and its combination with ASM was a perfect pairing

Outstanding results

Doom - 1993

- 2.5 D graphics
- BSP space partitioning for fast collision determination and rendering
- Ray Casting based rendering, lights

Outstanding results

Quake I - 1996

- First (!) real 3D graphics
- Polygon based models
- BSP space partitioning
- Lightmaps and real time lights
- Optimized for MMX instruction set - Michael Abrash

Outstanding results

Unreal Engine 1 – 1996 - 1998

- Real 3D graphics
- Polygon based models
- BSP space partitioning
- Lightmaps and real time lights
- Fog and well optimized

FUTURE OF SOFTWARE RENDERING…

Future of software rendering

- It was dominant until the first appearance of GPUs (1996)
- For a while, the two models ran parallel
  - Now, software rendering has almost completely disappeared from real time rasterization
- **The reason is the limit of the BUS speed:**
  - The CPU performance is increased heavily
  - Although it could not hold the race with the newly introduced GPU architecture
- **Today:**
  - almost every device take a GPU
  - GPS, smartphones, tablets, consoles, etc,
  - Their programming is relatively uniform: **OpenGL ES**

Future of software rendering

- **The rendering technology has not disappeared:**
  - Its role will again grow in the next few years
  - because of the increased number of cores in CPUs
- Three DirectX 9 compatible software renderer:

  1. **Pixomatic renderer** – RadGameTools
     Non free licence
  2. **Swiftshader** – TransGaming
     Free version is available

- Both product are well-optimized: take advantage of modern processors instruction sets (pl. SSE, AVX,MMX,3DNow)
- 3. **DirectX WARP** – Microsoft software library

The last great achievement

Unreal Tournament 2004 – Pixomatic renderer

UT2004 can be configured to run in software mode

GAME OVER