

*University of Derby*

*College of Engineering and Technology School of Electronics, Computing and Mathematics*

# 5CC510 - Graphics II

*Report for Graphics II - 5CC510*

100485878

March 2020

# CHAPTER 1

## Introduction

This report will compare a range of Graphics API, and the evolution of their key features and support for writing computer games. The report will also consider their latest versions, current implementations and potential for future development.

### The Graphics API

There are many graphics API used today, some having seen use and development since the early 90s. API, like programming languages can be seen as low or high level, some are cross-platform and some are designed for use in web-browsers, based on JavaScript.

This report will focus mainly on Direct3D, OpenGL & Vulkan, while also highlighting ideas drawn from Metal (Apple’s API for their iOS, macOS & tvOS) & Mantle (an API available to AMD Graphics Cards which led to the development of Vulkan).

Initially, video games like Quake (1996) were software-rendered, meaning they would run using the CPU to render and rasterize graphics. The problem with this is that there is a huge amount of mathematical calculation being done, which is where the GPU entered. Quake received a port in the following months supporting hardware acceleration via a very early chipset known as the Rendition Vérité; offering vast improvements like fuller colour, reduced pixilation, dynamic lighting and anti-aliasing. Not only would the game look better, but it would run better. Unfortunately, the port was tailored to this specific line of GPUs, offering no compatibility with other chipsets. As the GPU market would start to increase, no games creator would be willing to write separate code for every GPU using a different API on the market. John Carmack, the lead developer, began supporting non-proprietary (or open-source) APIs.

# CHAPTER 2

## API Development

### Evolution of Direct3D

DirectX came about because of Windows. As Microsoft moved away from DOS, which allowed access to all hardware and on to Windows, where access was restricted to these, there became a need to standardise a way of accessing the GPU to offload all heavy graphics computation to create better graphics.

It started off as Direct2D, part of the “Game SDK” on Windows 3.1 and started to gain popularity. At this time however, Microsoft was working on porting the rival OpenGL to Windows95 as it could handle 3D graphics. ‘This project failed but led to the development of Direct3D, which included a joystick control applet and support for Intel’s MMX technology, with later bugfixes.’ (Eisler, Craig's Musings, 2006)

DirectX 4.0 would contain logic for use in laptop video cards. However, with the latter being delayed, it never released. During or soon after this time, DirectX 5.0 was being developed with force feedback support for joysticks and gamepads, multiple monitor support, a new control panel for game controllers, a better User Interface and better MMX support which made its way into Windows 98.

DirectX 6.0 boasted greater performance than its predecessor, with bump mapping, texture compression and stencil buffers. These are methods still used in versions 20+ years later.

With further improvements to DirectX 7.0 and 8.0, it added pixel and vertex shaders, better lighting as well as increased performance, although was considered unpleasant to use. DirectX 8.1 addressed usability and became the basis for the XBOX API, from Microsoft’s collaboration with nVidia.

DirectX 9.0 released with a new version of the High Level Shader Language (HLSL) and shader model 2.0, MRT, an invaluable feature for real time shading and performance.

## Bibliography

1. Y. Achdou, F. Camilli and I. Capuzzo-Dolcetta, “Mean Field Games: Numerical Methods for the Planning Problem”, *SIAM Journal of Control and Optimization*, vol. 50, pp. 77-109, 2012. Available: 10.1137/100790069.
2. R. Albert, H. Jeong and A. Barab´asi, “Error and attack tolerance of complex networks”, *Nature*, vol. 406, no. 6794, pp. 378-382, 2000. Available: 10.1038/35019019.
3. F. Bagagiolo and D. Bauso, “Mean-Field Games and Dynamic Demand Management in Power Grids”, *Dynamic Games and Applications*, vol. 4, pp. 155-176, 2014. Available: 10.1007/s13235-013-0097-4.
4. A. Baraba´si and R. Albert, “Emergence of Scaling in Random Networks”, *Science*, vol. 286, no. 5439, pp. 509-512, 1999. Available: 10.1126/science.286.5439.509.
5. M. Bardi, “Explicit Solutions of Some Linear-Quadratic Mean-Field Games”, *Network and Heterogeneous Media*, vol. 7, no. 2, pp. 243-261, 2012. Available: 10.3934/nhm.2012.7.243.
6. A. Barron and J. Plath, “The evolution of honey bee dance communication: a mechanistic perspective”, *The Journal of Experimental Biology*, vol. 220, no. 23, pp. 4339-4346, 2017. Available: 10.1242/jeb.142778.
7. D. Bauso, “Consensus via Multi-population Robust Mean-Field Games”, *Systems & Control Letters*, vol. 107, pp. 76-83, 2017. Available: 10.1016/j.sysconle.2017.07.010.