# **Graphics and its impact on computer science**

By Billy Robinson

Contents

[Graphics and its impact on computer science 1](#_Toc36055468)

[How is this topic important within Computer Science? 3](#_Toc36055469)

[History of this topic within computer science 3](#_Toc36055470)

[Examples of this topic being implemented in computer science 3](#_Toc36055471)

[Summary of what I’ve learned from this topic. 4](#_Toc36055472)

[Reference List 4](#_Toc36055473)

# **How is this topic important within Computer Science?**

Graphics are important to computer science as they allow us to represent information given by a data source onto a computer scene: allowing us to visualize the data given. This can be useful in a wide variety of cases. Each graphic contains many points and vectors that allow us to form a completed graphic. A great example of graphics within computer science being used every day in the real world is computer games: they have characters and worlds with many points that are connected with vectors: often each character or scene will have thousands of these points and vectors to make a character look complete. These require a lot of processing power from the computer (in some circumstances even a separate graphics card). Graphics are used in a wide variety of different computer science disciplines, for example, mobile phones, computer games, and even computer displays.

Graphics allow us to view data in a way that is often easier to understand. From programs and documents, we often just have a load of numbers: creating graphics and using these to allow us to visualize data through various varieties of graphs. This means that we can process this information and understand it better as discussed by Slutsky, 2014. In addition to this, it also allows for easier explanation of findings during business meetings or other analytic based presentations.

# **History of this topic within computer science**

Graphics were first introduced by The Lumiere brothers’ in the year 1985 (Pruitt, 2020). They introduced the concept of graphics by creating movies, however – these were not computer-created graphics per se: instead of using mattes to create a similar effect. The first true computer-generated graphics didn’t become recognized until the 1950s with projects such as the SAGE air defense system (McConnell, 1995). It was mainly introduced by the united states military’s development within radar technology: as new kinds of display methods were required to present these with ease: computerized graphics were created.

Since the original creation of computerized graphics: technology has evolved at a fast pace. There are now many companies and researchers working towards creating effective solutions to many different types of data to allow them to be displayed on a graph and be as simple as possible for humans to interpret, as well as allowing for systems such as gaming systems to function correctly. A good example of a company that has made a big impact on the computer graphics sector is NVIDIA. In the year 2004, they created a graphical chipset that would allow for graphical rendering on a computer far more advanced than integrated chipsets (Houston, 2004), this allowed regular computers to do a lot more when it came to graphical rendering than was previously possible.

# **Examples of this topic being implemented in computer science**

<Image:
a basic ray traced model showing reflection and refraction.>Computer gaming is a major industry that computer science runs. It’s pushed along many advancements within graphics and graphical processing. Computer games utilize computer graphics by setting points and then connecting them, often extremely small connections that require a lot of rendering power to make smooth. Recently: it has created many advancements and helped to push them into the mainstream industry: one of which being ray tracing: which allows basic shapes to be used then use a light ray to make it look realistic. (Figure 1, Neil Dodgson, 2000)

Another important area where graphics are being used within computer science is within graphical user interfaces: within many various applications and websites, data must be given to the user who Is using the said application: this is often done within the usage of graphics (such as line graphs, bar charts, etc). This allows people to use the application to get a better sense of the data being presented to them: rather than just giving numbers – it allows people to visualize the data that they are given.

Graphics have also helped to create 3D models for real-world design problems, which has lead to the development of many programs based around creating modeling for engineering projects (programs such as AutoCAD). This makes the lives of people doing designing much easier as it means they don’t have to do as many calculations. It also allows for them to visibly show designs and mathematics to a client without having to explain the mathematical calculations that could be confusing.

# **Summary of what I’ve learned from this topic.**

I’ve learned a lot of transferable skills from this topic: one of the main skills that I’ve learned from this project is how to present data that is easy for a client to understand: it can often be easy to expect a client to have advanced knowledge of computer science: however, this often isn’t the case. As a result, simple to understand graphics must be put into place to ensure that we can explain how a piece of software works and how the data given from that piece of software can be analyzed using computer-generated graphics. I’ve also learned some valuable information on mathematics whilst studying this and this has acted as a refresher on some basic mathematics that I haven’t done for a while (e.g. Pythagoras theorem).

# 

# **Reference List**

* Pruitt, S., 2020. *The Lumière Brothers, Pioneers Of Cinema*. [online] HISTORY. Available at: https://www.history.com/news/the-lumiere-brothers-pioneers-of-cinema [Accessed 13 March 2020].
* McConnell, J.J. (1995). Computer graphics education: Issues from multiple perspectives. *Computers & Graphics*, 19(2), pp.331–334.
* Slutsky, D. (2014). The Effective Use of Graphs. *Journal of Wrist Surgery*, [online] 03(02), pp.067–068. Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4078179/>.
* Houston, M. (2004). *Parallel Rendering Workshop*. [online] IEEE. Available at: http://graphics.stanford.edu/~mhouston/VisWorkshop04/GraphicsClusters.pdf [Accessed 13 Mar. 2020].
* Neil Dodgson, (2000), Ray Tracing Example [ONLINE]. Available at: https://www.cl.cam.ac.uk/teaching/2000/AGraphHCI/AG/img/simplert.gif [Accessed 23 March 2020].

# **Counting and probability and its impact on computer science**

By Billy Robinson

Contents

[Counting and probability and its impact on computer science 5](#_Toc36055454)

[How is this topic important within Computer Science? 7](#_Toc36055455)

[History of this topic within computer science 7](#_Toc36055456)

[Examples of this topic being implemented in computer science 7](#_Toc36055457)

[Summary of what I’ve learned from this topic. 8](#_Toc36055458)

[Reference List 8](#_Toc36055459)

# **How is this topic important within Computer Science?**

This topic is important to computer science as it allows for computers to function on a very basic level: without the ability to count, predict probability or to have pseudo-randomness within programs (such as encryption) it means that we need counting and probability. Otherwise: this would lead to vulnerable programs and systems. Besides: it allows for developers to implement systems such as random choice-based systems and count the number of loops a program has completed, which is essential for things such as repeating loops for example ones on a timer that automatically check systems: an example being a computer checking the temperature every 10 seconds at a powerplant: this topic is essential to ensuring our infrastructure is reliable.

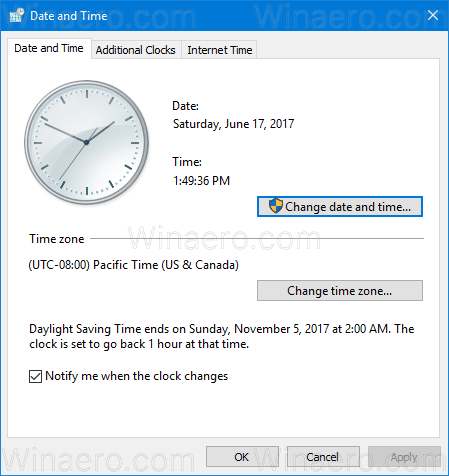
Another important contribution that this topic has to computer science is timing within devices: for example, things like computer clocks: computers run on this timing system. It allows us to regulate the speed of programs and allows us to determine how fast a computer runs. It also allows for things such as prediction algorithms to exist allowing us to power things such as artificial intelligence making choices as well as storing times within databases (will be discussed later).

# **History of this topic within computer science**

Probability has been something that had led to the modern development of mathematics. The earliest known examples of probability were developed between the 8th and 13th centuries (Ryding, 1998). This was done by finding all combinations of letters and vowels within Arabic. Determining the probability that one vowel was put together with another. Within computer science: we use a very similar mechanism within artificial intelligence: by calculating the probability that a piece of data meets specific standards through multiple layers of data analysis.

Within computer science: in the 1960s a time system known as UNIX time was created (Mashey, 2004). This time system counts the number of seconds since January 1st, 1970 and allows computers across various operating systems to compare the time with other timecodes, it also allows for things such as timezones and date formats to not get in the way whilst programming. It works as an effective solution to storing time within databases.

# **Examples of this topic being implemented in computer science**

The first example of this topic and how it’s been implemented within the computer science industry is Artificial intelligence: artificial intelligence relies on various stages of comparing data that we’ve given an algorithm to the new data that it’s been given: at each stage of this comparison it will compare with more data until the system comes out with a probability. For example: if we were trying to identify a car: we’d give the program “training data” (Moltzau, 2019) of pictures of cars, then give it a car and see if it can identify it as a car. This uses probability to determine whether it thinks it’s a car at each stage and what its confidence level is.

An example of this working is the windows clock: an extremely basic example which requires a computer to count then convert the counter to the current time. This works by using your computer's clock and determining it’s clock speed: The information gathered by calculating the clock speed is then passed to the clock discipline algorithm which uses the information collected to correct the current time on a computer (shortpatti, n.d.) as if the clock speed is to fast or to slow, over time it can lead to incorrect results: meaning programs that run off Windows’s time system will break.

Figure 1: Winaero.com, 2020

Another example of counting being used within the computer science industry is databases using Unix: a lot of databases will store timecodes using Unix: this allows them to convert a date into a basic number: allowing for easier comparison when it comes to dealing with this data within a program. Unix represents the number of seconds since January 1st, 1970 [source2] and means that it can be accessed without issues such as date formatting being an issue, as sometimes on different operating systems and in different locations, they format date and time differently. This allows them to have a unified time system to compare.

# **Summary of what I’ve learned from this topic.**

A major thing that I’ve learned from this topic is programming Skills: this topic has shown me information about how a computer processes counting within algorithms: for example, adding a number to a variable and how the computer processes and saves that data. This allows me to write more efficient code. With counting it also means I now understand how to do time conversions between different operating systems (for example – the same numbers being applicable on both windows and Linux).

From this topic: I’ve also learned a lot about critical thinking skills as it has allowed me to ask questions and do research into the topic which has allowed me to gather critical information about this topic and understand how timing as well as how prediction algorithms work: which has allowed me to gain a more significant and fact-based understanding of a wider variety of topics. One example of which being AI and how probability powers AI.

# **Reference List**

* Ryding, K.C. (1998). *Early medieval Arabic : studies on al-Khalīl ibn Aḥmad*. Washington: Georgetown University Press.
* Moltzau, A. (2019). *Artificial Intelligence and Training Data*. [online] Medium. Available at: https://towardsdatascience.com/artificial-intelligence-and-training-data-8020a8e2279c [Accessed 25 Mar. 2020].
* MASHEY, J. (2004). *Languages, Levels, Libraries, and Longevity - ACM Queue*. [online] queue.acm.org. Available at: https://queue.acm.org/detail.cfm?id=1039532 [Accessed 25 Mar. 2020].
* shortpatti (n.d.). *How the Windows Time Service Works*. [online] docs.microsoft.com. Available at: https://docs.microsoft.com/en-us/windows-server/networking/windows-time-service/how-the-windows-time-service-works [Accessed 25 Mar. 2020].

‌

# **Parallel Computing/Threading** **and its impact on computer science**

By Billy Robinson

Contents

[Parallel Computing/Threading and its impact on computer science 9](#_Toc36125681)

[How is this topic important within Computer Science? 11](#_Toc36125682)

[History of this topic within computer science 11](#_Toc36125683)

[Examples of this topic being implemented in computer science 11](#_Toc36125684)

[Summary of what I’ve learned from this topic. 12](#_Toc36125685)

[Reference List 12](#_Toc36125686)

# **How is this topic important within Computer Science?**

This topic is important because it allows computers to run multiple programs at once: which is extremely important to allow multiple applications to run on one system: Imaging trying to run one program per machine, it’d be extremely slow and would be extremely resourced intensive: it’d also mean that many modern computer systems wouldn’t run, as often they’re running on machines that have multiple threads to allow for multiple operations to run at once. It’s extremely impactful and important to have systems that run on multiple threads: not only does it allow multiple programs to run at once, it also allows for a singular program to spread its intensive load across many different threads, which can make for better user experience for the person using the program.

In terms of servers: It’d be borderline impossible to run only one service on a server with only one thread. Servers often use up many, many threads to run many different programs: the fact that we have threading and parallel computing allows us to run multiple programs and services within a server: meaning servers have a lot more potential power to run their services and perform for their clients/operators. This has meant that computer scientists have a lot more potential power on better-performing systems without being limited to doing on task.

# **History of this topic within computer science**

The first-ever system to support multithreading was the NBS SEAC in 1950 and DYSEAC in 1954 (Smotherman, 2007) this system supported only 2 threads which couldn’t work parallel to each other: meaning that one thread would have to run before the other thread could run. (GeeksforGeeks, 2018). However only a few years later a 33-thread system was developed, this was much more efficient and used multiple threads parallel to each other. This is much closer to modern systems as most modern systems run everything simultaneously with multiple threads: allowing them to run multiple programs without slowing things down or having to queue things and execute them after other threads. However: many developments occurred since this and have led to better systems being developed.

In modern times, on November 20th, 2000 (TGDaily, 2007) the Pentium 4 was released: which was the first processor to perform hyper-threading, which is currently a modern standard used in most computer systems. Hyperthreading allows for the processor to “split” itself into two separate processors, one thread is for the operating system and one thread is for the program that runs on this operating system. This allows a good balancing of programs over the computer and means that it’s not as slow. This is currently exclusive to Intel processors.

# **Examples of this topic being implemented in computer science**

Within modern-day computer science: most newly built windows machines use multiple threading technology which allows it to run multiple actions at one time without having to wait for other threads to execute. This depends on whether the manufacturer of the machine has used an outdated processor or a modern one. This capability allows programs, such as games to spread themselves across multiple threads. This can mean that rather than lagging and slowing down a gaming experience: it instead spreads the calculations and workload across multiple threads. This allows for a smoother experience and allows a machine to perform better than it previously would have.

Another example of this topic being implemented within computer science are server machines, server machines power the computer science industry as almost every major software system uses server machines to run off to allow things such as databases, websites to function and allow for a centralized network to run their systems off: because of this, it means the software would run much slower and have to queue to execute certain tasks.

# **Summary of what I’ve learned from this topic.**

From this topic: I’ve learned a deeper understanding of how computers work. Before learning about this topic I didn’t understand what threads are or that computer could originally only run one program without multiple thread capability, it’s extremely interesting to find more information about how computer used to run and how multiple threads can allow for a computer system to be more efficient and allow for a wider ability to cover demand. Another thing that I’ve learned from this topic is a deeper understanding of other operating systems: I typically only use windows as an operating system: whilst researching I did a lot of research on Unix based operating systems and how they used to only allow you to perform one task at a time: It’s extremely interesting to see the advancement of computer hardware over time and see the changes that have been made within computer hardware to make running multiple programs possible and parallel computing has been a driving factor behind this ability. I’ve also learned a lot about running multiple programs on one machine: this understanding has also helped me to understand how I can better code my programs to run on multiple threads have better performance: in the long run, I believe that this will help me to become a better computer scientist.

# **Reference List**

* ‌Smotherman, M. (n.d.). *Multithreading -- Mark Smotherman*. [online] people.cs.clemson.edu. Available at: https://people.cs.clemson.edu/~mark/multithreading.html [Accessed 27 Mar. 2020].
* ‌TGDaily. (2007). *Intel intros 3.0 GHz quad-core Xeon, drops Pentiums*. [online] Available at: https://tgdaily.com/33351-intel-intros-30-ghz-quad-core-xeon-drops-pentiums/ [Accessed 27 Mar. 2020].
* GeeksforGeeks. (2018). *Difference between Multiprogramming, multitasking, multithreading, and multiprocessing*. [online] Available at: https://www.geeksforgeeks.org/operating-system-difference-multitasking-multithreading-multiprocessing/ [Accessed 27 Mar. 2020].

‌