

Introduction

Way back in time in 1984, a school friend and I were busy writing games for the Commodore 64. With no internet, everything we learned came from books and various computer gaming magazines. We taught ourselves assembly and machine code and wrote support tools to allow the easy creation of music and sprites (graphical entities) for our games.

We then sent our games to various software houses such as Bug-Byte, Ocean, and Cable Software. We were invited in to chat with Bug-Byte in Liverpool and Ocean in Manchester. Following this, we spent a few weekends hanging out at Ocean in Manchester, which was a bit of a crazy place with teenage coders messing around with the latest computers, such as the Dragon 32 and BBC Micro. We were asked by Ocean to create a version of *BurgerTime*, but we decided to do our own thing and sent that to Cable Software in Luton, who then published the game. That was in 1984.

[Saboteur - Commodore 64 Game - Download Disk/Tape - Lemon64](#)

Fast forward to April 2025: after dropping out of an Open University MSc in Cyber Security, I decided to revisit writing games. That has now reached the point where I am about to self-publish a game written mostly in C++ using the Unreal Engine. This paper describes the approach I took, as an experienced C++ programmer, to becoming a semi-competent modern game creator.

Initial Decisions

The first choice I had to make was whether I wanted to use a game engine, and if so, which one to use. Although many games run on bespoke game engines (such as *Elite Dangerous* running on Frontier Developments' proprietary Cobra Engine) or on modified engines (such as the *Batman: Arkham* series), creating a new game engine or modifying an existing one would be a case of trying to run before I could crawl. For that reason, using an established game engine was a no-brainer; it simply came down to choosing which one.

Choosing a Games Engine

Before diving into examining different game engines, I created a simple list of criteria for what I wanted:

- **C++ Support:** Since I have years of experience in C++, this was a "must-have."
- **Current and Well-Supported:** Working with old or deprecated engines would likely become painful due to working around bugs and known performance issues.
- **3D Support:** Some engines do not support 3D, and among those that do, some handle it better than others.
- **Effectively Free:** Some engines are completely royalty-free, while others are royalty-free until a certain revenue threshold is reached.

Here are the engines that were considered but rejected:

Godot Engine

Pros:-

- Completely free with no subscription (MIT Licence)
- 2D and 3D support

Cons:-

- Although C++ is supported and indeed the engine is written in C++ the default workflow isn't to use C++
- Smaller asset store compared to other engines such as Unity and Unreal
- Standard 3D capabilities are no match for Unity and Unreal.

Unity Engine

Pros:-

- Free tier up to an annual revenue of \$200,000.
- Good 3D support

Cons:-

- While written in C++ game development is targetted at C#
- While Unity is capable of high visual quality this can be trickier to implement than on Unreal.

Finally I chose

Unity Engine

Pros:-

- Free tier up to an annual revenue of \$1,000,000
- Excellent 3D support
- Native C++ support
- Actively being developed. New releases around once or twice a year.
- Large support with asset library via the FAB market and huge numbers of developers using and producing supporting material for it.

Getting up the learning curve

Having decided on Unreal Engine, the three obvious ways to start learning were:

- Unreal's documentation and sample projects
- Free YouTube videos
- Paid books and video courses

Unreal's documentation and samples are good, but I felt I needed a more structured approach. I looked at a number of YouTube videos—many of them excellent—but again, I felt I needed a structured path to build a broad foundation in these technologies.

At the time, I still had access to the Open University library, which gave me access to O'Reilly books and videos, many of which covered the topics I needed. I decided to treat the exercise as if it were a college course with a number of modules. The areas I chose to study were:

- Creating an Unreal Engine Game from scratch
- Multiplayer Game Development with Unreal Engine
- C++ game development
- VFX and the Niagara effects system
- Blender and creating 3D models
- Creating movies

Having decided on the topics I found the following books and video

Unreal Engine 5 – Create A Zombie Survivor FPS Game (video course) - Ivan Yosifoc www.oreilly.com

This was essentially like standing behind a game developer for 6 hours while he made a rough prototype. It is unpolished and unedited—Yosifov jumps around, makes mistakes, and changes his mind on implementation details. However, as a basic introduction to what is possible, it was a reasonable starting place that introduced many core concepts within the Unreal Editor. It relies on Blueprints rather than C++ code.

Unreal Engine 5 – The Complete Begginer's Course (video course) – David Nixon www.oreilly.com

An excellent, highly structured course that walks you through making a game from scratch while explicitly teaching foundational concepts. Unlike the previous course, this provides a much broader education, using the sample game as a learning tool rather than just a build demonstration. By the end, you have a working simple game and a firm grasp of Unreal Engine basics.

Unreal Engine 5 – Multiplayer Game Development with Unreal Engine 5 (book) – Marco Sechi www.oreilly.com

Multiplayer in Unreal Engine is simultaneously straightforward and remarkably complex. Coming from a client/server architecture background, it takes time to adjust to having the same codebase run across both client and server. The book walks through first principles and provides solid examples to build a deep understanding of replication and networking.

Unreal Engine 5 – Elevating Game Experiences with Unreal Engine 5 (book) – Goncalo Marques, Devin Sherry, David Pereira, Hammad Fozzi www.oreilly.com

I selected this specifically because it covers C++ integration. As an introduction to programming Unreal in C++, it was excellent and filled with useful exercises. As a minor drawback, having four separate authors disrupted the prose rhythm occasionally, and as with many fast-moving engines, parts of the text can become slightly outdated over time..

Build Stunning Real-time VFX (book) – Hrishikesh Andurlekar www.oreilly.com

The Niagara VFX system is a fundamental part of Unreal Engine. This book guides you through step-by-step exercises to teach the underlying visual effects principles. After completing it, you can build custom effects and confidently adapt assets created by others.

Virtual Filmmaking with Unreal Engine 5 (book) – Hussin Khan www.oreilly.com

Unreal Engine can be used to make movies (see Rogue Trooper – Duncan Jones). A slightly off-the-beaten-path choice, but cinematic skills transfer directly to cutscenes and game presentation. The book takes you from basic editor navigation to producing a short animated sequence, enhancing the overall scene-building process.

Conclusions

My learning strategy was heavily influenced by traditional structured coursework and taking advantage of free academic access. It took roughly 4 to 5 months to read, watch, and absorb the materials before I felt ready to start building my own game. I initially estimated development would take 6 months, but it ultimately took 12. Much of that timeline expansion was due to early optimism and hitting pitfalls that a more experienced game developer would have known how to avoid.

After a year of hands-on development, my game, *Zombie Royal*, is about to go live on Steam.

Looking back, I could likely have reached a similar point using YouTube tutorials and official documentation alone. However, finding comprehensive C++ content on YouTube is harder, as C++ content is less likely to generate revenue compared to the more popular blueprint videos. Away from You-Tube platforms like Udemy offer extensive C++ courses (such as complete 60+ hour multiplayer C++ courses for reasonable prices), which are great alternatives.

[Online Courses - Learn Anything, On Your Schedule | Udemy](#)

If I were starting today, I would look closely at Udemy subscriptions or specific targeted video courses alongside official resources. Regardless, having completed my first "learn-on-the-job" title, I plan to continue studying new techniques and architectural approaches. The journey continues.

Mike Fielding

Software Engineer - Perihelion Software