Andrew Patton

SOFTWARE ENGINEER - BSc, MSc (ongoing)

andrewpatton.dev | atpatton97@gmail.com

[On Request] | 28 The Meadows, Fawdon NE3 3NA

UNIVERSITY GRADES

BSc (Hons) Computer Science with Games Development

Final Grade	**75% (1:1)**

MSc Advanced Computer Science

SEM 1	**81%**
SEM 2	**81%**
Research	**Ongoing**
Final Grade	**Ongoing**

TECHNICAL PROFICENCIES

- C#
- JavaScript
- Java
- Unity
- Unreal
- C++

These languages and tools are a short selection based on personal experience and preference.

HOBBIES & INTERESTS

- Podcasts about technology and physics (StarTalk, Vergecast)
- Comics (Spider-man!)
- Game Jams
- NBA – 76ers!

UBISOFT — **ONGOING**

User Research Lab

Evaluating not only the quality of a game, but reporting back what areas are lacking, and how they might be improved.

ACCENTURE — **JULY, 2015**

Work Shadowing

Working in AGILE/scrum teams. Executing & documenting tests, following a predefined plan. Exploring different work aspects of a team and how they interact to co-develop a software product

PROFILE

Technology enthusiast currently studying MSc Advanced Computer Science at Northumbria University. I help to run the NU Game Dev society; we meet up weekly, share projects/advice and participate together in game jams. Through the society and research conducted at university, I've grown to love not only programming but teaching others, seeing the spark in their brain when it all finally clicks.

NOTABLE PROJECTS

More projects can be found under the "Portfolio" section of my website.

The Broken Arms – Unity, C#, Mixamo

TBA is NUGameDev's most recent jam game, for 'repair' themed GGJ20. The player takes over a bar: serving customers, managing stock BUT everything keeps breaking! My focus was the AI customers, the complaints system and post-jam I took care of adding poo splatter, because, well, why not?

Software Architecture for Games (82%) – C++, DirectX

Using C++ to extend a DirectX wrapper into a functional 2D game engine, applying an optional advanced element. My engine was built on the concept of generic, flexible components, similar to the Unity framework. Gameplay programmers can add their own game objects, attaching both core and custom components to alter the behavior of and breathe context into the game object. This module focused on the underpinning programming patterns of a game engine, and the strengths/weaknesses of different approaches, along with a supporting commentary on how this can greatly affect the working environment of a large-scale development team.

Implementation of Object-Oriented Designs (92%) – C#, WinForms, EF

The focus of this module was not the program itself, but the quality of code and the thought process behind building an architecture that could withstand shifts in paradigms. Whilst the end-product was not too exciting, I found the development cycle and the module's lectures very useful & interesting.

Computer Networks and Control Systems (92%) – C, Java

Using a combination of C++ and Java to control a sprite with a custom peripheral (MBED board) across a local network (via ethernet), the on-screen program displayed a game inspired by Atari's 1979 *Lunar Lander*. The player must manage limited fuel resources whilst finding a flat area to land safely, controlling the craft using the board's joystick & potentiometers. Flashing LEDs and sound effects gave feedback to the player, simulating the ship's on-board systems (speedometer, altimeter, fuel gage, etc.)