Lee Griffiths

07843 119 789 lee@lee-griffiths.net

Available for work in Hertfordshire, Cambridgeshire or surrounding areas. **Willing to relocate**.

http://uk.linkedin.com/in/LeePGriffiths https://bitbucket.org/Poddster http://stackexchange.com/users/23045

Digital version of CV, containing *much* more detail can be found at http://www.lee-griffiths.net/cv/

Overview

Software developer. **7 years' experience** writing WDDM/**DirectX** 2D/3D **graphics drivers** in semiconductor industry for *Imagination Technologies*. Left in Aug 2015 for a career break, now looking to get back into work.

Worked on large-scale GPU projects, involving many HW and SW teams, targeting fixed and expensive silicon fabrication deadlines. Experienced full development life-cycle whilst writing **embedded, low-level, real-time code** for **drivers and firmware**, and **writing tools** for use by developers. Completely comfortable writing **non-driver code**: desktop apps, 3D apps, utility scripts. Targeted various OS (Windows, Linux, custom RTOS) on different form-factors (simulated GPUs, tablets, phones, desktop).

Can design, code, test and document software to a high standard. Take pride in personal and professional responsibility. Have a personal emphasis on promoting clear communication between teams to help avoid ambiguity and tribalism. And also on the testing and documentation of code to help ensure its **correctness**. Capable of working on **solo projects**, in teams, or as part of a **300-person effort** with **little supervision**. Capable of independent action to get information and work from other teams. Not scared to work on projects that many people rely on. Gained some experience in leading and supervising junior developers in their work.

Key skills: C, Python, graphics drivers, debugging IEEE 754 floating point (C, GPU shader code)

Professional Experience

Imagination Technologies, Kings Langley, Hertfordshire August 2008 - August 2015

Leading Software Design Engineer

Graduate S.D.E. (Aug 2008 – Aug 2010) S.D.E. (Aug 2010 - Jan 2015) Leading S.D.E. (Jan 2015 - Aug 2015)

Market leading IP (intellectual property) company. Owner of brands: **PowerVR, MIPS**, PURE, Ensigma. PowerVR GPUs used by many leading-brand smart devices (e.g. **all Apple iPads & iPhones**), PS Vita, TVs, Laptops, etc. DirectX driver shipped on Windows-based laptops and tablets, and helped in validation and verification of all shipping GPUs.

- Worked as **a senior member** of the WDDM team (~30 people), writing WDDM/DirectX 9, 10, 11 drivers for Windows XP/Vista/7/8/10. User-land and kernel mode C.
  - All levels of the software development life-cycle, notably a driver that started as a **0-file project** and was delivered to customers *(Intel, Allwinner*) with a **full WHQL pass** on Windows (Vista, 7, 8.1), complete with maintenance periods, on Intel x86, x64 and ARM based platforms.
  - **Mentored** new/junior engineers. Helped train, educate and present DirectX topics to SW/HW engineers. Advised HW teams on DirectX spec requirements and **signed-off** some hardware specs.
  - Helped indirectly **support customer** by providing technical answers in tickets (IPGear, TeamTrack). Have been "responsible engineer" for WDDM/DX team when Product Managers are taping-out a core and driving all bugs to 0, or for when a customer's testing/validation fails on a DirectX test.
- Responsible for **design, architecture and implementation** of large sections of the DirectX driver codebase:
  - Majority of work in user-land DirectX driver dealing with the "**shader** stack", e.g. shader program compilation, execution, GPU data/code memory management.
  - Other areas of lead responsibility: DirectX 11 tessellation, GPU command stream/packet formation, constant buffer allocations, code that read "apphints" from windows registry and .ini files (in both user and kernel drivers), code that dumped shader log files. Supervised DirectCompute implementation.

- Had significant input into design and architecture of other areas of the driver. Worked on every file in the WDDM/DX driver codebases in a fire-fighting/debugging fashion.
- Worked on many **useful software tools** that were used in the WDDM team and PowerVR wide. E.g. Python scripts for: C-codegen, translating between source-control systems.
- Regular contributor to the **vital tool** "objanal": a fast GPU simulator and important driver debug tool used by almost all HW/SW/Sim engineers in PowerVR (~300 people)
  - **Performance sensitive** code due to important validation/test system turnaround times.
  - Helped code review all patches for objanal, once a code review system was in place.
  - Contributed to design and architecture discussions.
  - Objanal written in C. Produced a .dll (or .so) that was driven by various frontends.
  - Contributed to frontends: command line (Python), graphical (C#/DirectX/OpenGL).
- Helped innovate the WDDM/DirectX team by introducing more **modern and efficient working practices.**
  - Helped introduce a peer **code-review system** into the team, also pushed for PowerVR at large.
    - Helped implement *mandatory* code reviews for all WDDM/DirectX code via **ReviewBoard**.
    - Acted as 'back stop' to ensure every single commit to the WDDM/DirectX codebase on ReviewBoard was reviewed by someone, usually myself.
  - Helped introduce, implement and maintain a **continuous-integration regression test environment** for the driver. Thousands of small-scale tests that all ran at the touch of a button, rather than manual testing via a few, very complex games & 3D apps.
    - A **dll extension to Python**, using the Python/C API, for DX10 API. A set of Python scripts and 'shell' environment, using comtypes and ctypes libraries, for DX9 and DX11 APIs. Allowed 3D test-apps to be written in Python, rather than in C or C++ using COM/DirectX.
    - Wrote hundreds of DirectX 9, 10, 11 3D test apps. Majority in Python, few in C or C++.
    - Helped develop & maintain QA/testing tools: an automated, **distributed**, multi-threaded tool for test scheduling and execution; test-image comparison and error reporting tools; an email delivery report system, system configuration parsing scripts. All in Python.
    - Maintained report hosting website. Django (earlier PHP) for dynamic content generation.
  - Continually pushed for improved working practices at all organisation levels (e.g. code reviews, better source control, improved documentation, change project organisation etc).
- Debugged/Reverse-engineered DirectX API output of many 3D apps, games & game engines to fix faulty images. Found and debugged many **IEE 754 floating point** problems in various forms e.g. generated shader code, HW FPU designs, and purely software FPU implementations.
- Found, triaged, debugged, proved and reported **many hardware bugs** in simulated-, prototyped- and live-hardware designs, all of which would have been **very expensive** to fix if not found.
  - Designed and implemented efficient driver workarounds for any unfixable or "won't fix" HW problems.
- Worked with various build systems for driver and tools. nmake/msbuild/Visual Studio.
- Understood the maths involved in creating 3D and 2d graphics and used this to write driver code and 3D apps. Helped educate junior team members in terms of related maths problems.

APT group, University of Manchester June 2007 – September 2007

Summer Vacation Student

Started work on the course materials that the APT group would use for a new second-year course module. Work continued on into third year project. For more details, see *UoM final year project (below)*.

PEVE group, University of Manchester June 2006 – September 2006

Summer Vacation Student

Created a new website for PEVE group to match "new style" template on CS department's webpages (cs.man.ac.uk).

Academic Qualifications

2005 – 2008	**University of Manchester, School of Computer Science**	B.Sc. Computer Engineering (Honours), Class 2:1
2002 – 2005	**Cardinal Newman College, Preston**	A-Levels in Computing(A), Maths(B), Environmental Science(B), General Studies (C), Accounting(D)
1997 – 2002	**St. Bede's High School, Lytham**	11 GCSE grade A-C

School of Computer Science, University of Manchester

August 2005 - July 2008

In addition to the usual CS topics (algorithms & data structures, databases, AI, etc) there was a particular focus on: Micro-controllers, Digital design of hardware (circuits, CPUs and systems on chip); Operating system design; Programming language theory & compilers

Example of Academic Projects

- In Java:
  - Simple multi-threaded servers: e.g. telnet, ftp, a "hotel booking" system.
  - Client software to book slots from a server running a booking database.
  - FAT12, process scheduler, virtual memory implementations.
  - MPEG audio encoding software intended for mobile devices.
- Developed a compiler & interpreter for an artificial, C-like language (lex, yacc, C)
- All of the ARM, PIC and C code from the Third Year Project
- Developed an I^2C (I2C) peripheral for an AT91 (ARM7TDMI) in Verilog. Synthesized onto Xilinx Virtex FPGA.
- Developed "Stump": An ARM-like 16-bit RISC processor in VHDL. (Only simulated, too big for FPGAs available in CS department at the time).
- Full custom CMOS layout for a ripple carry adder (Cadence suite)

Final Year Project

Hand-picked by University staff. Created a hardware platform/system to be used by the CS department for new 2nd year course. Worked with one other student (only joint project of that year!). Large freedom of design: Single requirement was that it used two specific boards, one ARM and one PIC based. Asked to design a complete system and sketch feasible applications that could be run on it, with the course changing the app used each year. Delivered technical report (~20k words) on the project as final year dissertation. Also produced handover report documenting the structure and function of the code/system for the course organisers. Gained experience of the full development life-cycle, taking it from a basic spec to a full working system.

The course apparently used the platform and codebase, with modifications, for a few years with no problems.

Hardware Platform Final Design

Hardware platform with a camera module that could be rotated around two axes. PIC board controlled rotation via stepper and brushed-DC motors. The ARM board worked in tandem with a VDEC1 video decoding co-processor to manipulate camera images, and used an FPGA to help accelerate certain image decoding operations. ARM/PIC/VDEC1 all talked over I^2C. Working example provided to staff was simple colour space conversion. A *theoretical* example application for the platform was motion detection & automated tracking.

Technical Skills used in the project

- Developed a basic RTOS on the PIC for a student's "client" application to use.
- Developed various drivers for both PIC and ARM: Motor control (stepper, brushed DC), communication (I^2C, RS232 and custom protocol), Character LED, FPGA interface/downloading, basic user-space stack-tracer for debugging.
- Programmed: PIC18, AT91SAM9 (ARM9), video decoder (VDEC1/ADV7183B), flash EEPROM, Xilinx Spartan 3.
- Developed the custom build process and tool chain for the project & course-module. Mix of Bash scripting, C programs (Linux) and some ARM assembly.
- Experience with Microchip's MPLAB software for programming & debugging the PIC.
- The project's software was written in C (PIC), PIC assembly, ARM assembly, and tools developed for Linux in C. The image processing/colour-space conversion running on a Xilinx Spartan FPGA was written in VHDL.

Personal Projects

Super-Sleuth https://bitbucket.org/Poddster/super-sleuth

Prototype for procedurally-generated detective game featuring the "realistic" simulation of people, their emotions and crimes. (~4k kloc Python). Documentation and design for full game underway.

Programmable Flight Computer for Kerbal Space Program https://bitbucket.org/Poddster/ksp_pfc

Unfinished mod for Kerbal Space Program. Attempt at "full system" computer emulation inside of KSP. CPU emulation mostly complete, but needs a vigorous re-factoring. Circuit simulation was in research + design stage. LLVM backend planned, not yet implemented. ~12kloc lines of C#, ~4.5kloc of test programs written in ksp_pfc assembly.

More projects available on https://bitbucket.org/Poddster

Technical Skills

Programming languages, technical skills, and technology use

Advanced Use	C (c89, c99), DirectX 9/10/11, HLSL (shaders)
Everyday Use	Python, IEE 754 (C & GPU HW), Drivers, WHQL/WHCK/WLK, concurrent programming/data access
Frequent Use	Assembly (x86, x86-64, ARM, GPU), Direct2D, doxygen, nmake (Microsoft version), msbuild, CMake, MinGW, MSys, SDL (via both PyGame and C API), C#, C++ (C++98, C++03)
Infrequent	Java, c11, OpenGL 4.5, GLSL 4.5, Bash scripts, gdb, POSIX + Gnu/Linux utilities (grep, sed, awk, etc), Win32 API, COM, XNA 4.0 (+ MonoGame), HTML 5.0, CSS 3, Event Tracing for Windows (ETW), MISRA C rules
I used this once to do something useful and still vaguely remember how to use it	Android, DirectX12, Vulkan, OpenCL, Pic18 assembly, Perl, MATLAB, MPLab, Apple's Metal API, AMD's Mantle API, SystemC, Ruby, Verilog, VHDL, Modern C++ (C++11 C++14), PHP, Oracle SQL & MySQL, LLVM backend + LLVM language ref, Lex, Yacc

Tools use

Advanced Knowledge	git, Pix For Windows
Configure/use	Mantis bug tracker, ReviewBoard, CVS, SVN, Mercurial (Hg)
Everyday Use	Windows, Linux, Visual Studio, WinDBG (kernel + user-mode), Perforce, MKS, Microsoft Project (Gannt chart), RenderDoc, Serena business mishaps (formally TeamTrack), CA Clarity PPM, Openproject, Bugzilla, Eclipse, Jenkins, Microsoft connect, Microsoft Driver Verifier,
Used a Few Times	GPUView, PVRTune, Intel VTune, ApiTrace, OllyDbg, Cadence Tools (Schematic & CMOS Layout editors. NC-Verilog/NCSim & Verilog XL simulators), Mentor Graphics (Schematic), SoftICE, XPerf, LaTeX

Hardware skills

Micro-controllers	All PowerVR GPUs from MBX to Rogue (series 3-7), parts of PowerVR Series 8, some PowerVR video & display cores, Intel Poulsbo (GMA 500), Intel Atoms, AT91SAM9621 (ARM9 based), AT91M40800 (ARM7), PIC18LF452, Xilinx Virtex & Spartan FPGAs, Analog Devices ADV7183B (Digilent VDEC1 video decoder chip). Some time with Arduino and Raspberry Pi.
Peripheral Hardware & Protocols	PCI bus, I2C, UART, USART, RS232, SPI. Analogue and digital PIOs. Interrupts, Power Management controllers etc. Motor control, Character LCDs. Watchdog/One-shot timers, 555 timers, etc.