Tom Murray

contact@freefunction.co.uk | +44 07500 016864
Bristol, UK | TomMurray | tompaulmurray |

Summary

Software engineer with 8+ years industry experience primarily with C++. 3+ years porting AAA Windows games to other platforms, contributing to several published works and leading development of one. 4+ years compiler and algorithmic design & development for a novel hardware platform aimed at machine learning applications. Now contracting, most recently developing Candy Crush Soda at King.

Available for C++-based development contracts & consultation. Open to all types of projects and have great experience in particular with desktop applications/UI, soft real-time applications (e.g. games), compilers, and algorithmic development & optimisation (incl. hardware acceleration).

Work Experience Overview

Senior Software Engineer - King (Contract) May 2023 - Dec 2023

- Contracted as a senior software engineer in the monetisation team for Candy Crush Soda Saga, one of King's biggest games.
- Worked in cross-functional teams (UI, UX, production, engineering etc.) to deliver new monetisation features in the game as well as bug-fixing and maintaining monetisation features. Wrote mainly in C++ and used King's in-house engine targeting Android, iOS, and multiple desktop & web platforms.
- Alongside 2 other developers and over ∼4 months successfully delivered a complete overhaul of the in-game shop, a key monetisation feature in the game.
- Sole developer for 2 other minor monetisation features from pre-production to delivery.

Software Engineer - Graphcore Feb 2018 - Jan 2023

- Member of the team responsible for the Poplar compiler/runtime, and PopLibs libraries. Responsibilities include: bugfixing, implementing new features, functional & performance testing, run-time performance optimisations for IPU hardware.
- Technical leadership & consulting for Poplar & PopLibs.
  - Design consultant for proposed Poplar/PopLibs APIs and features.
  - Advised members of the team (and external contributors) on implementing new features & worked with them to fix bugs.
  - Modelled whole application and kernel-level performance/memory usage as collateral for hardware design decisions, or evidence of value of hardware features proposed by the software team.
  - Presented several internal technical talks about Poplar/PopLibs design & implementation.
- Tech lead for a new team responsible for design, planning, and implementation of an unannounced compiler project since the beginning of 2022. Project makes heavy use of MLIR tooling.

Developer - Feral Interactive Oct 2015 - Feb 2018

- Project Lead on macOS/Linux ports of Mad Max from original Windows source code written largely in C++ and using DirectX. Led the project from receiving first source code drop to shipping Oct 2016.
- Developer on 3 other ports of Windows games: BioShock Remastered macOS, BioShock 2 Remastered macOS, Dawn of War III macOS/Linux.
- Last 6 months focused on development of Apple Metal code paths in core libraries. Performance optimisations, debug/workaround of graphical issues, implementation of new features. Liaison with third parties for new features and driver issues.

Placement Student - Feral Interactive Jul 2013 - Aug 2014

- Contributions to several ports of AAA Windows games to macOS.
- Implemented first shipped version of downloadable content management in Feral games that allowed users to purchase and download downloadable content for 4 Feral titles during my placement year and several more titles since then.
- 6 months developing a macOS port of GRID 2 from Windows from first delivery of source code. Shipped one month after finishing my placement.

Education

2011 - 2015 BSc w/ Honours, Computer Science, **University of Bath** (First Class)

Skills

Fluent languages	C++ (98, 11, 14, 17), C (99 mostly)
Competent languages	Python, Lua
Technologies & Libraries	MLIR, Clang, Boost, Intel Thread Building Blocks (TBB), OpenGL, OpenCL, Apple Metal
Tooling	Git, Subversion, gdb, lldb, Intel V-Tune, vim, Xcode, CMake, CTest, GoogleMock, GoogleTest, GitHub, Phabricator, Jenkins, Confluence, Jira

Selected Projects

MLIR-based compiler project @Graphcore

Acted as the technical lead for an ongoing unannounced compiler project within Graphcore for ∼1 year. This project used the LLVM project's MLIR tooling to create a new intermediate representation and set of transformation passes to perform optimisations at different levels of abstraction from IPU hardware.

Responsibilities included: technical design/planning/direction, software architectural design, implementation (C++ 17/TableGen), liaison with other technical leaders, being a point of contact for technical questions from outside of the team, providing guidance to team members, and keeping up to date with MLIR/LLVM project discussions, current practices, new features, and general direction to feed into the project.

Top-K & Sort Implementations @Graphcore

Code — Documentation

Working alone, I created an optimised implementation of top-k & sort based on the bitonic sorting algorithm over the course of 7-8 weeks. I adapted bitonic sort to implement top-k, and to the distributed memory architecture of Graphcore's IPU. I wrote kernels to run on individual processors using C++17 & assembly (custom ISA) with coordinating code in C++ 17.

The top-k operation in a key target application for the new implementation was ∼48x faster than with the previous heap-sort based implementation, resulting in the application as a whole running ∼2x faster. This implementation continues to be used today in a wide range of applications.

Poplar Code Overlays API @Graphcore

Overlays — Documentation

Working alone, I was responsible for designing & implementing an experimental overlays feature for IPU hardware in the Poplar compiler & run-time. After meeting with internal customers for this feature, I designed and presented the "FunctionBuffer" API. I then implemented the first version of this API which allowed code to be stored off-chip and loaded during execution of a program.

This functionality was successfully used to enable internal customers to run programs that did not otherwise fit into limited on-chip memory by swapping out code from off-chip memory during run-time.

Sparse Fully Connected Layer Implementation @Graphcore Code — Tech Note

Over the course of ∼4 months, myself and a colleague were responsible for designing and implementing a library for a sparse fully connected layer in neural network where weights for the layer are sparse. We designed and implemented a custom algorithm for IPU hardware.

We split work roughly 50/50. I personally was responsible for part of the design & specification, implementation of some of the kernels in C++17 & assembly (custom ISA), some of the coordinating code in C++17, and later for writing a tech note as documentation/a kind of code companion (linked).

The aim of the project was to drive a customer engagement by providing the functionality they required while operating at or close to a pre-calculated theoretical maximum throughput for IPU hardware which we delivered.

Max Max macOS Port @Feral Interactive Ltd. Game wiki

I was lead developer of the macOS port of Mad Max from receiving the original C++ 11 source code targeting Windows/DirectX through to the final release over the course of 1 year. For several months I worked alone, and for the remaining time additional developers joined me to work on a Linux port and bring the game to release quality.

Some of my contributions were: making the project compile and run in Xcode (and its version of Clang), adding missing features to in-house implementations of Windows/DirectX/other third party APIs using macOS/OpenGL/OpenCL APIs and more, debugging/fixing initial major issues (spanning functional, graphical, and performance issues) such that the game was playable before putting it through several cycles of QA, and performance optimisations to reach the widest possible range of target hardware.

The project successfully met QA requirements and released in October 2016.

Downloadable Content Management/UI @Feral Interactive Ltd.

Working alone, with regular feedback, I delivered a system that could be used in any Feral game to manage DLC (downloadable content) for their games. This included a UI for users to purchase, download & manage DLC implemented in C++ with the Carbon UI framework for macOS. The backend for the Mac App Store SKU was implemented with the Objective-C StoreKit framework. For the Feral Interactive store SKU the backend used HTTP requests to custom built web APIs, with an in-house product key based system for ownership verification.

Several titles during and after my time at Feral were released with this system which drove sales of DLC and also allowed for independent delivery of optional content such as language packs as free DLC to reduce the base game size on disk.