Peter James Austin
263 New Hey Road, Birkenhead, Wirral, CH49 9BS; Email: peteaustin1993@gmail.com
Mobile: 07809203559; Github: `https://github.com/petejaustin/`

I am currently a PhD student at the University of Liverpool, studying Infinite Graph Games with Perfect Information, Automata Theory, and Formal Verification. During the academic year for my masters and my PhD studies, I taught first year BSc Computer Science students discrete mathematics, algorithmic complexity, video game design, automata theory, and object-oriented programming. I have also taught masters year Computer Science students about verification techniques. Mathematics and algorithm design are some of my strongest industrial abilities; having contributed to several nuanced codebases that required experience with more theoretical concepts.

Education

University of Liverpool : 'Games, Automata, Verification' PhD studies **2022-Present**

Studying two player infinite graph games, namely the computational complexity of reachability games bounded by a strict equality of some constant timestep T encoded in binary. It is conjected that this is PSPACE-complete.

Currently implementing a novel algorithm to solve two-player Mean Payoff Games (MPGs). This algorithm in question is the `Dorfman-Kaplan-Zwick` algorithm, published in ICALP 2019. This algorithm has an upper-bound time complexity of $O(min(mnW, mn2^{n/2}logW))$, making this the fastest deterministic algorithm to determine the winning regions of both players in an Energy Game or an MPG. This is being written in C++, and uses CMake as well as Makefiles for build automation of the whole game solving library it is being integrated into.

Designed a Parity Game plugin for a graph visualisation tool called Workcraft (`https://github.com/workcraft`). This was built in Java, and can be used as a pedagogic tool for people to quickly model Parity Games in a drag-and-drop environment. This plugin uses algorithms based off of Zielonka's recursive parity game solving algorithm due to its efficient average case time complexity.

University of Liverpool : Advanced Computer Science with YEAR IN INDUSTRY MSc Distinction **2020-2022**

Modules covered: Advanced Algorithmic Techniques, Privacy & Security, Research Methods in Computer Science, Technologies for E-Commerce, Data Mining & Visualisation, Machine Learning & Bioinspired Optimisation,
Safety & Dependability, Computational Intelligence

- Produced an academic analysis of a plethora of computational algorithms for the MSc end of year project. Presented a comprehensive breakdown of such algorithms that can be extended to engineer solutions for many computational problems, both practical and theoretical

University of Liverpool : Computer Information Systems BSc - 2:1 **2015-2020**

Modules covered: Complex Information Networks, Computational Game Theory, Knowledge Representation & Reasoning using Logic, OS Concepts, Database development, Advanced OO programming, Advanced Software Engineering, Distributed systems, Internet principles, Software development tools, Applied database management, Principles of C & memory management

Cisco Networking Academy : CCNA **2013-2014**

- Self-taught how to create and maintain both client and enterprise networks

City of Liverpool College : BTEC **2012-2014**

- Triple Distinction* (D*D*D*) in Extended Diploma IT Level 3

Work Experience

University of Liverpool **2020-Present**

Demonstrated and tutored first year BSc students discrete mathematics, set theory, and propositional logic during first semester. Taught Year 2 BSc students about algorithmic complexity, including concepts such as time and space complexity, asymptotic notation, and basic data structures (stacks, queues, heaps). Dynamic Programming and recursion were large portions of the curriculum. Taught MSc students about verification focussed topics such as model checking, Markov chains, Markov decision processes, and Hoare's calculus.

Alongside teaching algorithmic complexity, video game design was taught from a pragmatic standpoint. This involved giving the students an introductory glance into using Unity, Unreal Engine 4, and JMonkeyEngine to develop independent video game projects. Tutoring included not only structured group sessions several times a week, but also 1-on-1 tutoring sessions in which students are given remedial help with their curriculum as and when necessary. Automata Theory and Object-Oriented Programming were also modules that were taught.

Defense Science & Technology Laboratory **2021-2022**

12 months as a software and electrical engineer at Dstl. Role involved producing software engineering solutions that may work on several different pieces of hardware such as radio systems, embedded systems, and servers. Main languages worked with while here are C++, C, Python, Rust, and Golang. Became well versed with using Git version control software to produce scalable and readable solutions within the department.

Outside of software engineering, several presentations were produced to give to shareholders that transparently display technologies the department have made. These may include more theoretical and mathematical concepts, as well as more hardware focused solutions that deal with the likes of Internet of Things and mobile development. Worked with Software Defined Radio technologies at Dstl, namely OpenCPI in order to develop transmission applications.

Directly contributed to low-level codebases using the C language that are currently being utilised for embedded technologies, facilitating communications within an unorthodox manner. Was placed in charge of unit testing this codebase, resulting in professional automated testing alongside the software product.

Leverhulme Research Centre for Functional Material Design **2021**

Worked closely with Franka Emika Panda robotic arms, developing scripts to program them to complete tasks such as probing and transporting test vials via impedance control laws. C++ was used to engineer source code that gave low level instructions to the robotic arm via the libfranka library provided by Franka Emika, and Python was used to give more abstract instructions such as traversing through joint or Cartesian space.

Libraries utilised were built using ROS (Robot Operating System) middleware as a direct dependency. Forward kinematics and inverse kinematics were used to efficiently translate commands given through joint space to Cartesian space as well as vice versa. For thorough testing of robotic scripts, both a simulated environment in Gazebo and a physical Panda arm inside the laboratory were used.

Hobbies and Interests

- Experience with Python, C++, C#, C, Java, R, Bash, MySQL, Git, OpenCPI, Go, Rust
- Languages: Russian (Conversational), Spanish (Basic) , French (Basic)
- Experience with using the Linux distributions Arch, Mint, Ubuntu, CentOS, and Kali. Deeply interested in completing coding challenges in my spare time (Advent of Code, Leetcode, Codewars, etc)

References available on request