Adrian Yip

587-966-9062 | adrian.yip@mail.utoronto.ca | LinkedIn | GitHub | Portfolio

Computer Engineering student writing performance-focused software in **C/C++** seeking a **12–16 month PEY** SWE role.

Education

University of Toronto — Toronto, ON
BASc, Computer Engineering + PEY Co-op — *Sep. 2024 – Apr. 2029 (Expected)*

Technical Skills

Languages: C++, C, Python, GLSL, RISC-V Assembly
Technologies: Vulkan, OpenGL, PyTorch, MediaPipe
Tools: Git, CMake, Valgrind, Linux

Experience

Undergraduate Researcher | DGP Lab — Sep. 2026 – Present
Dynamic Graphics Project, University of Toronto — *Toronto, ON*

- Contributing to an audio-driven **3D talking avatar** for psychiatric-interview training, built on a speech → **FACS blendshapes** → **FLAME** face model → **GAGAvatar** rendering pipeline.
- Implemented **real-time rendering** at **20 fps** that drives GAGAvatar from FACS-driven FLAME motion, streaming frames in sync with speech audio as they are generated (**Python**, **PyTorch**).
- Integrated **MediaPipe** iris tracking to drive the avatar's FLAME eye pose from the user's webcam, so the avatar holds eye contact with the user in real time.

Website Developer and Tournaments Director | uoftbadmintonclub.com — Sep. 2025 – Present
University of Toronto Badminton Club — *Toronto, ON*

- Built and deployed the club website serving as the information channel for **2,500+ members**.
- Directed tournaments for top players, in charge of scheduling and sponsor coordination for **$5,000+** prize pools.

Projects

Living NPC | *C++, Vulkan, GLSL* | GitHub | Video — Sep. 2026

- Placed **semifinalist** at Hack the North 2026 out of **1,045 participants** by building a custom **Vulkan/C++** rendering engine and math library (enginemath), for animating interactive 3D NPCs.
- Loaded and animated glTF characters with a **custom model loader**, skeletal skinning, and morph target blending, driving 52 facial blendshapes per NPC through a procedural lip sync system with coarticulation rules and a conversational gaze state machine.
- Applied two **SIGGRAPH** papers (JALI, S3) to build a procedural animation pipeline where a single string of text produces synchronized lip movement, eye gaze, and body motion with no motion capture or pre-recorded clips.

OpenGL Shader Renderer | *C++, OpenGL, GLSL, CMake* | GitHub | Video — May 2026 – Jun. 2026

- Developed a multi-pass deferred renderer in **OpenGL**, chaining off-screen G-buffer and post-processing framebuffers to render multiple non-photorealistic styles (cel-shading, isophotes) on Assimp-imported meshes.
- Built gl-profiler, a reusable profiling library that times each render pass on the GPU with OpenGL **timestamp queries** and a recycled query pool, allocation-free per frame.
- Cut bloom GPU cost **4×** after profiling identified it as the slowest pass, dropping separable Gaussian blur to quarter resolution with bilinear upsampling, holding **60+ fps at 1600×1200** on integrated **AMD Radeon 610M**.

OpenStreetMap Route Planner & Courier Optimizer | *C++, OpenStreetMap* | Video — Jan. 2026 – Apr. 2026

- Built a Google-Maps style navigation app over **OpenStreetMap** data that searches streets, intersections, and places, and provides a route between any two intersections with an estimated travel time.
- Implemented **A*** for point-to-point pathfinding and **Dijkstra** to precompute traveling times for the Traveling Courier Problem, optimized with simulated annealing and a multithreaded greedy heuristic, achieving routes **within 6% of the best solutions** among 64 teams.
- Verified performance and zero memory leaks under **Valgrind** with **90+** functional and performance tests on full city datasets.

Pokemon RPG with Bare-Metal C | *C, NiosV DE1-SoC, VGA* | GitHub | Video — Jan. 2026 – Apr. 2026

- Built a complete Pokemon RPG from scratch on bare-metal **RISC-V** (DE1-SoC FPGA, no OS, engine, or libraries), spanning overworld exploration, battles, catching, and a Pokedex.
- Drove the game with a 36-state FSM and direct register-level hardware I/O, achieving **zero heap allocation** by storing immutable assets in compile-time static lookup tables and game state in static buffers.
- Designed a **data-driven** table decoupling species, moves, and type charts from combat logic.