Alex Xu

Phone: +1 (647) 674-3251
Email: alex.zihao.xu@gmail.com
GitHub: https://github.com/AlexZihaoXu
Portfolio: https://me.alex-xu.site/

Skills

Languages:
Python, C/C++, C#, Java, JavaScript/TypeScript, HTML, CSS, Markdown, Lua, Dart

Framework / Library / API:
React, Electron, Flask, SocketIO, Django, Unity, Pygame, GLFW, LWJGL, SFML, Google Cloud Platform, Oracle Cloud, Firebase, OpenAI API, numpy, Discord.py

Tools:
IntelliJ Idea, PyCharm, CLion, VSCode, Typora, Gradle, CMake, SSH, Blender, Fusion 360, Crua.

Awards

Canadian Computing Competition (CCC)

- Grade 9 Junior (75/75)
- Grade 11 & 12 Senior top 25%

YR Hacks

Hackathon held by York Region District School Board

- Grade 10: Best Use of Cloud Computing
- Grade 11: Best Game

CanHack

CanHack by the DMZ at Ryerson University

School 2nd Place

Education

University of Waterloo

2023 - current

Currently pursuing a Computer Engineering degree at the University of Waterloo.

Middlefield C.I.

High school (2019 - 2023)

Graduated from Middlefield Collegiate Institute in the year 2023.

Projects & Experience

Game Dev

Basic Minecraft Implementation with Java *(Java, JOML, LWJGL, OpenGL, GLSL, stb)* [Dec 2021 - Jan 2022]

- Implemented a custom rendering engine using **OpenGL** completely from scratch, achieving 3-4 times framerate boost in comparison to other frameworks such as **JMonkeyEngine**.
- Lighting system with **BFS**, supporting anti-aliasing and multi-threaded terrain generation.

Minecraft Remake in C++ *(C++, CMake, GLFW, OpenGL, GLSL, glu, stb)* [Sept 2022]

- Written in **C++**, with **50% lower** memory usage. Tested with **Valgrind** to ensure no memory leaks exist.
- Custom rendering engine with **OpenGL** as the rendering pipeline.

Party Physics Game *(Java, Java2D, Dyn4J, Socket)* [Dec 2022 - Jan 2023]

- Utilising **OpenAL** for the sound engine, supporting surround effect.
- Custom active rendering framework based on **Java Swing**, with anti-aliasing feature.
- Implemented multiplayer feature on top of custom networking framework with **TCP** socket. Implemented an abstraction layer to serialize and deserialize objects with minimum processing time. Was able to handle more than 300 objects with a latency under 10 ms.
- Real-time physics simulation using **Dyn4J** library, synced over network with barely noticeable latency conflicts thanks to the custom networking framework.

Frontend

Personal Portfolio Website *(React, TypeScript)* [Jan 2024]

- Utilized the **React** library to build my personal portfolio.
- Hosted on an **Oracle Cloud Ubuntu** server with **Nginx**, utilizing HTTPS with **Certbot** for secure communication

Backend

Decide4Me Backend *(Python, Flask, Firebase)* [April 2021]

- Developed an MVP within 24 hours and won the "Best Use of Cloud Computing" Prize at YRHacks.
- Collaborated seamlessly with a frontend developer.
- Used **Flask** to build a RESTful API deployed on **Google Cloud Platform**. Utilized **Firebase**, for data storage and user authentication, ensuring scalability and real-time data synchronization.

Hardware

Robotic Arm Project *(STM32, C, Fusion 360, Cura)* [Oct 2023 - Nov 2023]

- Implemented the project from scratch using the **STM32 Nucleo** board. Programmed the entire system and UI in the **C language** to achieve optimal control and performance.
- Modeled the entire robotic arm in **Fusion 360**, optimizing the design for 3D printing with **5 iterations** for the project.
- Custom-designed gear set incorporating mechanical advantages to enhance torque & energy efficiency.
- Implemented an **I2C LCD driver** with a scheduler that runs asynchronously, and configured **ADC DMA** joystick to prevent the main process from blocking.

Others

Minecraft Launcher with mods syncing *(Java, Python, Flask, Processing)* [March 2023]

Developed a cross-platform Minecraft launcher in Java.

Client Side:

- Utilized **Processing** for UI components and rendering.
- Implemented automatic file synchronization by fetching file lists from the server API, ensuring up-to-date and complete mod installations.
- Integrated a player list retrieval feature from the server API, eliminating the need to launch the game to access player information.

Server Side:

- Implemented the server using **Python** with **Flask**.
- Deployed using **Docker** and hosted on an **Ubuntu** system on **Oracle Cloud**, with a custom domain registered through **Cloudflare**.
- Created a mod enabling real-time player list export on the server.