BSU

Shkurdyuk Roman

Phone
+375299829397

Email:
romansh0106@gmail.com

GitHub
romanshkurdziuk - Overview

PROFILE

2nd-year student at the Faculty of Applied Mathematics and Computer Science (BSU). I specialize in C++17 development with a strong focus on system programming, multithreading, and application architecture.

I have experience building both high-load backend solutions (REST API) and complex graphical applications (Qt 6). I possess a deep understanding of memory management, algorithms, and design patterns (MVC, State Machine, Observer).

1. TECHNICAL SKILLS

- Languages: C++17 (STL, Smart Pointers, Lambda), Java (Core).
- Frameworks & GUI: Qt 6 (Core, Widgets, Graphics View, Multimedia), Crow Framework.
- System Programming: Windows API, Multithreading (std::mutex, std::thread), IPC (Named Pipes), Memory Management.
- Architecture & Logic: OOP, Signals & Slots, Finite State Machine (FSM), Event Loop.
- Tools: CMake (Advanced configuration), Git, GoogleTest, VS Code.

2. PROJECT PORTFOLIO

2.1 Multi-threaded REST API Service (To-Do Backend)

Stack: C++17, Crow, CMake, JSON

A backend service for task management supporting concurrent requests.

- Implemented MVC architecture and a full CRUD API.
- Ensured Thread Safety for data access using mutexes.
- Developed a custom persistence system (File Storage) with crash recovery capabilities.
- Configured automated dependency management via CMake FetchContent.

2.2 2D Space Shooter Game Engine (Qt 6)

Stack: C++17, Qt 6, QGraphicsView, CMake

A full-featured 2D shooter with modular architecture and procedural generation.

- Architecture: Designed the system based on the QGraphicsView Framework, decoupling logic (Game Loop) from presentation (View) using the Observer pattern (Signals & Slots).
- AI & Logic: Implemented a Finite State Machine (FSM) for complex enemy behavior (attack/wait phases) and a wave spawn system (WaveManager).
- Performance: Optimized sprite rendering and memory management (RAII, deleteLater) to prevent memory leaks during active object creation/destruction.
- Mechanics: Implemented precise collision detection (shape(), collidingItems), parallax effects, and a resource manager (.qrc).

2.3 Multi-threaded Client-Server System (WinAPI)

Stack: C++, Windows API, Named Pipes

A database management system utilizing low-level OS mechanisms.

- Designed inter-process communication via Named Pipes.
- Solved the "Readers-Writers" synchronization problem using Critical Sections and Events.
- Implemented dynamic real-time updates of binary data files without server downtime.

2.4 Industrial-Grade Math Library

Stack: C++17, GoogleTest

A library for mathematical computations.

- Achieved 100% Unit Test coverage (GoogleTest).
- Implemented type overflow protection (std::overflow_error) and optimized memory allocations via reserve.

3. ALGORITHMIC BACKGROUND

Actively applying complex data structures and algorithms in projects:

- Data Structures: Segment Tree, DSU (Disjoint Set Union), BST, HashMap.
- Graphs & Search: Dijkstra’s Algorithm, DFS/BFS, Connected Components search.
- Optimization: Dynamic Programming (DP), Huffman Coding, Greedy Algorithms.

4. EDUCATION

Belarusian State University (BSU)

Faculty of Applied Mathematics and Computer Science (FAMCS)

- Major: Applied Mathematics
- Year: 2nd Year (2024 – Present)

5. SOFT SKILLS

- Problem Solving: Skilled in resolving "Dependency Hell" and refactoring legacy code.
- Code Quality: Committed to writing reliable code using strict compilation flags (-Wall -Wextra -Werror).
- Adaptability: Self-taught Qt 6 and WinAPI to a level sufficient for building functional systems.
- English level: B1 - B1+