Daniil Muratov

Junior game developer

Bratislava, Slovakia (on-site or remote) • cyber.younge@gmail.com • +421 903 117 362
github.com/jaimsalbert650-max • linkedin.com/in/daniil-muratov-gamedev • jaimsalbert650-max.github.io/cv

I build the rules of a game as a separate, testable layer and put the presentation on top of it. My lane-defence game runs its whole economy and combat simulation in TypeScript, with 24 test files that execute the rules in Node without a browser, and it ships both as an HTML5 web build and as an Android package. I write the numbers down before I write the code, the economy, the difficulty curve, the payout rules, then rewrite them when playtesting disagrees. Two solo games finished from design document to a playable build, plus an action RPG prototype built around a stat and modifier system. Ukrainian and Russian native. No commercial experience yet, looking for a first role in a studio.

259 COMMITS / 4 WEEKS | **101** C# FILES | **37** TS FILES | **24** VITEST FILES | **17/17** NUNIT PASSING

CORE COMPETENCIES

Game mathematics and economy design. I specify systems as numbers before implementing them: currency income against unit cost, XP curves and level-up pacing, damage and stat formulas with damage types and multipliers, difficulty ramps per wave. Balance is tuned against real playtests, not guessed once and left.

Testable architecture. Simulation kept independent of the renderer so the rules can run headless. 24 Vitest files covering economy, combat and board rules; 17 of 17 NUnit EditMode tests on the Unity prototype, where the Core layer is plain C# with no engine dependency.

Unity 6 and C#. Gameplay logic in C#: enemy behaviour and states, wave spawners, weapon and projectile behaviours, pickups, progression, object pooling, damage and health. Android builds. Assembly definitions to enforce layer boundaries.

TypeScript and HTML5. TypeScript with Phaser 3, Vite for the web build, Capacitor for Android packaging. 37 TypeScript sources across board, wave loop, economy, combat and UI.

Design documentation. I write the design document before the code: systems, economy, difficulty curve, balance numbers, and I rewrite it when playtesting disagrees. UI specified as a design pack with shared components and design tokens, treated as the specification rather than a sketch.

Git and AI-assisted development. Daily Git practice, small self-describing commits, branches per feature, 259 commits on a single project over four weeks. Daily work with **Claude Code**: specification and plan first, implementation and review after.

Unity 6 | **C#** | **TypeScript** | **Phaser 3** | Vite | Capacitor | Vitest | NUnit | Git | Assembly definitions | ScriptableObject | HTML5 | Android | Claude Code

PROJECTS

Food vs Pests

solo project, 2026

TypeScript · Phaser 3 · Vite · Capacitor · Unity 6

- One design, two implementations: a TypeScript build on Phaser 3 shipping as an HTML5 web build and an Android package, and a second implementation of the same design in Unity 6 that builds to an APK.
- Simulation separated from rendering, so 24 Vitest files run the economy, combat and board rules in Node with no browser involved.
- Designed on paper first: a board of 5 lanes by 16 cells of depth, a sugar-cube economy with income and unit costs balanced against each other, an enemy roster, and an explicit loss condition.
- Mousetrap safety net redesigned after playtesting, with written rules for burrowed enemies, coin payout and enemy broods, because the first version made the loss condition unreachable.
- UI written as a design pack before it was built, with shared components and design tokens, and treated as the specification during implementation rather than as a sketch.
- github.com/jaimsalbert650-max/Food-vs-Pests

Dinner Rush solo project, 2026

Unity 6 · C# · Android

- Complete run loop in C#: timed waves, several enemy types, auto-firing weapons, pickups, and an XP and level-up flow where the player picks one of several upgrades.
- Weapons written as distinct behaviours rather than damage numbers: a pasta twister that vacuums a crowd in and throws it back, meatballs that fall slowly and land as a splash, a sauce puddle that burns what stands in it.
- 101 C# files split by domain into Core, Enemies, Weapons, Player, Progression and UI, with object pooling for projectiles and enemies.
- Built and tested on a physical Android device. 259 commits over four weeks.
- github.com/jaimsalbert650-max/Dinner-Rush

Isometric action RPG solo project, in progress

Unity 6 URP Mobile · C#

- Stat and modifier system with damage calculation across damage types and factions, the part of an RPG that is pure mathematics, kept in plain C# and unit tested.
- Layered architecture enforced by Unity assembly definitions, App above Gameplay and UI, above Data, above Core, with Core free of any engine dependency.
- Event bus, service locator and state machine. Content described as ScriptableObject data: classes, enemies, items, skills and loot tables, so numbers can be retuned without touching code.
- 17 of 17 NUnit EditMode tests passing.

EDUCATION

2026 – present **Self-directed game development**
Unity and C#, TypeScript and Phaser, design documentation, Git and testing, learned by building complete games rather than exercises. Bratislava, Slovakia.

2025 **Secondary school leaving certificate**
Ukrainian secondary education, 11 years. Ukraine.

LANGUAGES

Ukrainian native • **Russian** native • **English** A2 • **Slovak** A2

English at A2 is enough to read documentation and technical English. Spoken English is weak and improving. I work comfortably in Ukrainian and Russian.

Based in Bratislava. Available immediately, full time or part time, on-site or remote. References available upon request.