GAME DEVELOPMENT SINCE 2007

Jes Obertyukh

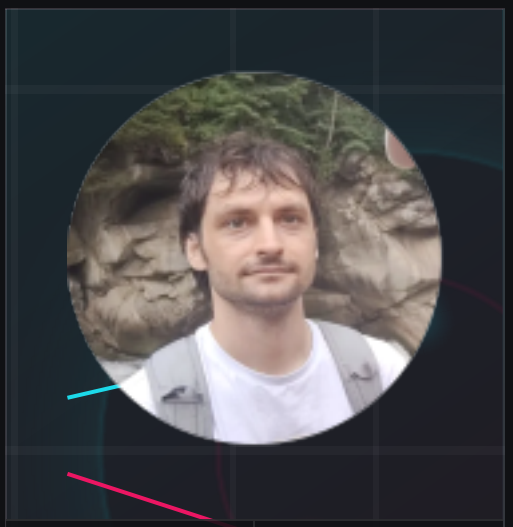
Senior / Staff Unity Engineer

Game Architecture • Networking

Rendering & Tech Art

I design game architecture, build networking, gameplay, and rendering systems, and optimize performance for mobile and PC games in Unity.

View LinkedIn ↗



~20 years in game development	**14+** years in architecture & leadership
Unity since version 3.5	**C#** since version 1.1

Contact

jesobertyukh@gmail.com

linkedin.com/in/jes-obertyukh-20528a50 ↗

Telegram: @JesObertyukh ↗

Time zone: CEST

Languages

Ukrainian	Native or Bilingual
Russian	Native or Bilingual
English	Professional working proficiency

Selected career highlights

8 built · 4 evolved

Designed core architecture for eight games and substantially improved four existing architectures.

CTO & Technical Lead

Led technical direction for **4** games as CTO at D.A.W.Studio. Co-led teams of **30+** people, including **10+** programmers, aligning development across programming, QA, and Tech Art.

64-player Battle Royale

Developed and optimized networking, gameplay, and rendering for Combat Master.

<10 → 30 FPS

Improved Combat Master's Battle Royale map performance on Xiaomi Mi A2 with runtime depth-aware impostors.

Core Skills

- Unity Game Architecture
- C#, C & C++
- Rendering Architecture: Built-in & URP
- Technical Leadership & Mentoring
- Real-Time Multiplayer Networking
- ECS, DOD, Unity Jobs & Burst
- Technical Art Engineering
- Technical Design & Game Rules
- Metagame & Backend Systems
- Voxel & Procedural Systems
- Shader Graph / HLSL / Cg
- Architecture Audits & Project Rescue
- Reliability & Fault Tolerance
- Editor Tooling & Automation
- CPU/GPU Profiling & Optimization
- Git, Tools, CI/CD & Content Pipelines

The whole game One technical system

For nearly 20 years, I've connected game architecture, networking, gameplay, and production

In lead, CTO, and consulting roles, I design and implement game architecture from initial foundations through production support, connecting gameplay, networking, rendering, content, and QA. I turn game-design intent into testable systems, lead technical decisions, and mentor engineers and technical artists.

ARCHITECTURE

Game lifecycle & system boundaries	Designed the lifecycle from initialization through gameplay, transitions, reconnects, and shutdown, with explicit service ownership. Separated simulation from presentation and defined server-only, client-only, and shared responsibilities
Reusable gameplay architecture	Developed a common application structure for single-player and server-backed games, reusable gameplay and UI systems, and an ECS-based gameplay foundation
Architecture recovery & iteration speed	Audited game flow, dependencies, testability, and hot paths; restored system boundaries and reduced coupling without unnecessary rewrites, removing integration barriers that slowed feature development

ENGINEERING

Gameplay & tools	Developed and optimized AI, pathfinding, voxel destruction, and reusable gameplay systems; built editor tools for level design, game configuration, and content validation
Metagame & backend	Delivered player profiles, quests, and purchases as complete features spanning client-side UI, game logic, and corresponding backend services. Used Protocol Buffers and MessagePack for communication with metagame servers
Server infrastructure	Built services with ASP.NET Core, MongoDB, and Redis, including matchmaking and battle-server orchestration
Metagame & battle flow	Designed transitions between metagame and battles with offline operation and seamless reconnect

NETWORK

Architecture & ownership	Owned the design, implementation, and production support of server-authoritative networking, including client/server contracts, Photon Fusion systems, and a custom layer over Unity Transport
Replication & serialization	Designed packed-buffer protocols, delta-compressed world snapshots, network ticks, and redundant input transmission; optimized field ordering for delta compression and handled out-of-order packets and MTU limits
Simulation & network performance	Implemented movement interpolation and optimized prediction, reconciliation, lag compensation, and input latency in projects using Photon Bolt and Fusion
Traffic & interest management	Optimized replication for 64-player mobile Battle Royale, prioritizing updates by view direction, distance, and aiming
Reconnect & resilience	Designed fault-tolerant reconnect for custom networking and extended existing networking stacks to restore interrupted sessions
Diagnostics & performance budgets	Defined tick rates, resimulation limits, and per-subsystem frame-time budgets. Added network data to Unity Profiler, analyzed reconciliation, and tested with up to 40% packet loss, high jitter, and 400 ms ping

8 built · 4 evolved

Designed the core architecture for eight games and substantially improved existing architectures in four others.

1–2 sec

Lifecycle architecture enabled direct startup into the required gameplay or test state.

2–3 days

Average delivery per engineer for a gameplay feature or complex UI screen, enabled by my architecture across four D.A.W.Studio projects.

> Good architecture should ensure that the fast solution is also the correct one

From art assets to the final frame

I design graphics systems, translate art direction into rendering solutions, and optimize Unity games for production

I connect rendering architecture, shaders, and content workflows to meet visual goals within hardware constraints. I extend Unity's Built-in and Universal Render Pipelines, including custom URP Renderer Features and passes, and work with artists and programmers on lighting, materials, meshes, batching, overdraw, and bandwidth.

1.6–1.8M → ~500K

Visible polygons in full-map views, preserving distant detail.

<10 → 30 FPS

Combat Master Battle Royale map on Xiaomi Mi A2.

~52 / <100 batches

Batches: ~52 on standard maps; below 100 on Battle Royale.

TECH ART

Content standards & validation	Defined and documented source-to-Unity standards: model formats and hierarchy, vertex attributes, UVs, model/UI texture sizes and padding, import settings, and repository organization. Combined manual review with scene and UI validators. Defined biome-content constraints upfront to limit asset dependencies and enable efficient world streaming
Animation architecture	Designed additive and override animation layers to support gameplay requirements with a minimal set of animation assets
Art pipeline automation	Automated import settings, texture previews, and low-resolution texture generation for streaming; built level-design validation checks
One-material map pipeline	Built a material-library editor tool that unified each map under one material and texture arrays, improving batching and reducing GPU memory bandwidth usage while making duplicate or overly similar textures easy to identify
Technical-art mentoring	Mentored artists across four studios, teaching how and why to choose rendering solutions: DCC-to-GPU geometry, vertex splitting and ordering, triangulation, mesh attributes, batching, shader-stage decisions, and camera/content-dependent cost. Two mentees later became discipline leads

RENDERING

Graphics technical leadership	In lead roles, owned rendering decisions, task estimation and breakdown, and code review. Often served as the team's primary rendering specialist, implementing solutions directly or giving actionable feedback on problems and proposed fixes
Implementation & maintenance cost	Evaluated implementation, integration, and ongoing support costs, favoring simple solutions and investing in more complex systems when they reduced longer-term production cost
Render pipelines & image quality	Built single-camera URP rendering with custom passes and Renderer Features, including a sniper scope with controlled render ordering and overdraw. Optimized water and caustics rendering and lightmap UVs; developed shader techniques that preserved distant detail
Shaders & visual effects	HLSL/Cg and Shader Graph custom code for stylized lighting and shadows, vegetation sway, atlas-based rendering, impostors, materials, and VFX. Built radial fog that outperformed Unity's linear fog in the target project, and optimized mesh rendering using compressed vertex attributes
World streaming	Implemented voxel-world chunk streaming and occlusion-based checks to avoid loading invisible chunks
Runtime depth-aware impostors	Replaced conventional LODs and distance hiding, preserving full-map detail without visible transitions within a 40–70 MB impostor texture budget
Profiling & frame debugging	Unity Profiler, Frame Debugger, Rendering Debugger, RenderDoc, Metal Debugger, and Snapdragon Profiler
Mobile GPU & platform compatibility	Profiled across iOS and Android GPU families and resolved severe shader performance issues on iOS. Kept shader code shared where practical, adapting logic for OpenGL ES and handling texture format and compression differences between mobile and PC

03 / ARCHITECTURE IN PRACTICE

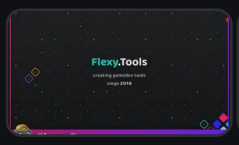
Experience, materialized

FLEXY.FUN

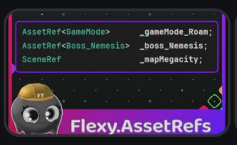
A reusable Unity architecture shaped by nearly two decades of engineering experience, built for modularity, testability, and fast iteration

Flexy.Fun is my long-term initiative for general-purpose Unity game architecture. Flexy.Bricks / ToYs defines the methodology; Flexy.Framework implements it through reusable systems, templates, and packages.

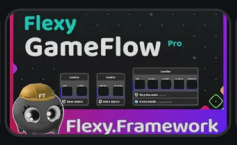
Flexy.Bricks / ToYs	An original architectural methodology refined since 2010 for modularity, testability, and fast iteration. Each unit runs independently in dedicated development and test scenes for implementation, debugging, manual verification, and automated testing
Flexy.Framework	A modular Unity framework defining composition, ownership, initialization, communication, and testing, with clear module boundaries and predictable flow while leaving game-specific design unconstrained
Implementation & delivery	Designed and implemented reusable systems as separate Unity packages, published on GitHub and the Unity Asset Store



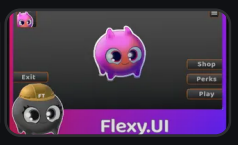
Flexy.Tools



Flexy.AssetRefs



Flexy.GameFlow



Flexy.UI

04 / WORK TOGETHER

Roles, consulting & mentoring

I'm open to:

- **Senior / Staff Unity Engineer or Game Architect roles** — broad hands-on ownership across gameplay, networking, systems, tools, and production
- **Senior / Lead Unity Networking Engineer or Network Architect roles**
- **Senior / Lead Unity Rendering Engineer or engineering-focused Technical Artist roles**
- **Independent consulting** — architecture audits, project rescue, and technical mentoring

From programmer to technical leadership

Mar 2025
Aug 2026
1 yr 6 mos

CUBIC GAMES

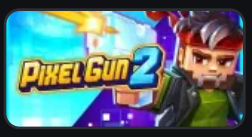
Game Architect & Staff Unity Engineer

Owned networking and core client/server and gameplay architecture for Pixel Gun 2, a voxel-based multiplayer FPS using Unity and Photon Fusion

- **Architecture** — Designed the full game lifecycle from initialization through metagame, coregame, transitions, reloads, reconnects, and shutdown, with services and dependency injection organized around GameContexts
- Decoupled game modes from maps and enabled direct Play Mode, host, and client startup from gameplay maps with synchronous frame-0 initialization
- **Render & TechArt** — Documented biome-content constraints for the art team to limit asset dependencies per map and prevent costly downstream memory and streaming problems
- **Network** — Owned Photon Fusion configuration and production troubleshooting; optimized prediction, reconciliation, lag compensation, input latency and precision, and gameplay/voxel-destruction synchronization. Implemented custom movement interpolation over Fusion
- Built Fusion simulation and presentation pipelines with server-only, client-only, and shared boundaries, using ECS/DOD principles and message-bus communication
- Trained the Pixel Gun 2 team in network coding rules, prediction approaches, and when to use each
- **Other** — Optimized voxel collapse and destruction-driven NavMesh rebuilding
- Modernized supporting systems with C# 10, FEPM, on-demand Addressables loading, and a flexible audio subsystem
- Defined network tick rates, resimulation limits, and frame-time budgets by subsystem, using profiler markers to identify excessive costs
- Optimized low-end mobile performance with algorithms, Jobs, Burst, and DOD; introduced ECS/NetCode-ready boundaries without a project-wide rewrite



Pixel Gun 2



Pixel Gun 2 on Steam

Mar 2023
Mar 2025
2 yrs 1 mo

ALFA BRAVO INC

Game Architect & Staff Unity Engineer

Developed and optimized gameplay, Photon Bolt networking, rendering, and content-delivery systems for Combat Master, a multiplayer FPS with 64-player Battle Royale matches

- **Architecture** — Designed the complete Content Update architecture covering asset layout, bundle versioning and assembly, and delivery through Google Play Asset Delivery
- **Render & TechArt** — Built a runtime depth-aware impostor system that increased full Battle Royale map performance from below 10 to 30 FPS on Xiaomi Mi A2, reducing visible geometry from 1.6–1.8M to ~500K polygons
- Built a texture-array material library and optimized meshes and attributes, reducing mobile GPU bandwidth while rendering standard maps in ~52 batches and Battle Royale in under 100
- **Network** — Worked across the Photon Bolt networking stack, optimizing lag compensation, client prediction, and reconciliation
- Implemented networked moving platforms and a networked train, alongside vehicle systems
- **Other** — Moved content-bundle assembly out of the monolithic build script into a modular task-based pipeline shared by local and CI builds
- Implemented perks, fully rebindable gamepad input, a player-profile editor, and ScriptableObject client-side tamper protection
- Continued supporting the team through periodic consulting on architecture, technical art, reusable Unity systems, and scalable backend solutions suitable for a team without dedicated backend specialists



Combat Master: Season 5



Combat Master: Welcome to Combat Zone!

Jun 2018
Mar 2023
4 yrs 10 mos

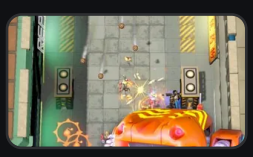
D.A.W.STUDIO

Chief Technology Officer

Led four programmers, technical artists, and project managers in a studio of about 20 people. Owned technical direction and technology selection; worked closely with the game design lead to align cross-department communication and workflows from design through implementation, QA, and delivery. Remained hands-on in engineering

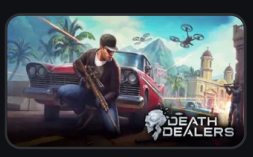
Designed and built from scratch the architecture for four games: two online multiplayer sniper games, a runner, and a roguelike shooter

- **Architecture** — Developed MC-VmV, a rethinking of MVVM that unifies application structure across single-player and server-backed games
- Built the entire coregame on an ECS-based foundation using Unity Entities, spanning movement, weapon firing, ballistics, controllable bullet-sprites, network synchronization, and other gameplay subsystems
- Built a ScriptableObject-based Game Design Information system that translated gameplay rules into structured, reusable configuration models, giving designers clear control over gameplay systems
- Built reusable client-side systems for player profiles, UI windows, asset bundles, battle quests, and game-designer-friendly IAP
- Enabled direct startup into the required gameplay or test state in about one to two seconds
- Developed an architectural approach that enabled each engineer to deliver a gameplay feature or complex UI screen in two to three days on average across four projects. Content and configuration became the primary production constraints
- **Render & TechArt** — Kept complete game scenes within roughly 50–100 rendering batches through rendering, shader, and technical-art optimization
- Optimized water and caustics rendering for an island map while preserving its visual quality
- Built a single-camera URP pipeline using Renderer Features, including a sniper-scope renderer combining zoomed and unzoomed regions with controlled render order, camera shifts, one camera cull per frame, and minimal overdraw
- Improved lightmap UV layouts to preserve quality at low atlas resolutions; developed shader and rendering techniques that kept road markings crisp at shooting distances beyond one kilometre
- Adjusted UI art production to achieve hardware-assisted antialiasing with negligible runtime cost
- Owned studio-wide asset pipelines from source formats and repository organization through builds and delivery. Defined texture import and padding, mesh and UV standards, and biome constraints to prevent cross-biome asset dependencies
- Designed additive and override animation layers to meet gameplay requirements with a minimal set of animation assets
- Coached artists in rendering performance and the reasoning behind technical production choices
- **Network** — As sole networking owner, designed, configured, and maintained a server-authoritative layer over Unity Transport for both multiplayer sniper games: packet and replicated-state formats, network data types, packed-buffer serialization, delta compression, world snapshots, network ticks, and redundant input transmission
- Optimized serialization so much of the data could be copied directly into buffers; handled out-of-order packets and kept datagrams within MTU limits, accounting for proxy headers to avoid fragmentation
- Designed the communication architecture spanning ASP.NET Core metagame services and UDP battle servers, with offline operation and seamless reconnect; the same resilience model was later adopted independently by Combat Master and preserved player continuity during a prolonged backend outage
- Designed battle-server orchestration, built a Python supervisor for managing processes on battle-server machines, and owned backend scaling decisions
- Built an ASP.NET Core player-profile service and matchmaker backed by MongoDB and Redis
- **Other** — Built a reproducible CI/CD pipeline around a Unity-owned build process, allowing developers and the build server to produce the same builds with the same configuration
- Built level-design and pathing tools with integrated animation and camera controls
- Selected and hired key specialists; helped strong mid-level and senior engineers strengthen their technical judgment and take ownership of senior-level decisions. One became CTO at his next company
- Built a Scrum delivery workflow in Jira and Confluence linking specifications, implementation, QA-owned acceptance, and documentation. Used relative complexity, historical velocity, and Value/Time prioritization for planning; combined User Stories, Job Stories, and JTBD with testable game rules and evaluated implementation, integration, and maintenance costs
- Partnered with game designers on story and gameplay across all four projects, translating design intent into consistent rules and resolving conflicts between features
- Simplified UI/UX by reducing unnecessary clicks and screen clutter while making interaction rules more consistent and learnable


Cyberstrike


Animals Happy Run


Sniper League: The Island



Death Dealers

Dec 2016
Jun 2018
1 yr 7 mos

TELLURION MOBILE GAMES

Lead Unity Developer

Improved performance and gameplay foundations for RealmCraft, a released mobile voxel sandbox

- **Render & TechArt** — Optimized CPU-baked voxel chunk lighting and rendering, shaders, culling, and radial fog while minimizing fragment-shader work; helped design delta updates that add or remove one light's contribution instead of recalculating every light in the affected area
- Implemented voxel-world chunk streaming with occlusion-based checks to avoid loading chunks known to be invisible
- Built working RealmCraft VR versions for Gear VR and HTC Vive, optimizing rendering and implementing VR-specific camera movement and controls
- Coached artists in rendering performance and content construction, explaining technical trade-offs and working together on suitable solutions
- **Other** — Designed extensible mob AI and accelerated pathfinding
- Rebuilt voxel-world movement for stability, performance, and customization



RealmCraft with Skins Export to Minecraft

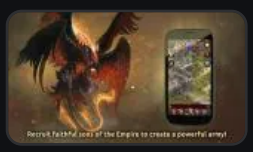
Jun 2016
Dec 2016
7 mos

SERIOUS CAKE

Lead Unity Developer

Led the Unity client for Age of Phoenix: Wind of War, a 4X strategy game backed by Akka.NET

- **Render & TechArt** — Improved runtime performance, reduced build size, and streamlined UI production for faster feature iteration
- Built an editor tool that exposed existing Unity Texture Importer dimension controls to designers, helping them choose more efficient imported texture sizes and reduce unused atlas space
- Coached artists in rendering performance and the reasoning behind technical production choices
- **Network** — Contributed selected Akka.NET backend features while retaining primary ownership of the Unity client
- **Other** — Conducted technical interviews and mentored programmers and technical leads



Age of Phoenix: Wind of War

Jan 2015
Jun 2016
1 yr 6 mos

WARGAMING.NET

Lead Unity Developer

Led client development for REVOLT across architecture, client/server systems, gameplay, and optimization

- **Architecture** — Rebuilt and stabilized the client architecture, restoring development flow and establishing reusable foundations for subsequent features
- **Render & TechArt** — Architected the battle-map rendering path to run reliably on iPad 2, using explicit object ordering to minimize overdraw while preserving batching
- Developed custom stylized shaders, lighting, shadow receiving, and shadow casting to support the hand-drawn visual direction
- Established a collaborative technical-art workflow for the game's visual style; mentored the artist in rendering performance, content construction, and the reasoning behind technical choices
- **Network** — Built a universal, fault-tolerant client/server command flow for simultaneous-turn gameplay, ensuring both clients replayed the same authoritative outcome
- **Other** — Worked in a mentor-guided Scrum team with rotating responsibilities and continuous improvement through retrospectives, later applying those practices as CTO



REVOLT Tank Battle

Sep 2014
Dec 2014
4 mos

PLARIUM

Technical Lead

Developed coregame and tools for Motorplanet, a mobile multiplayer racing game

- **Network** — Implemented car movement synchronization and interpolation for multiplayer racing, with smoothing to mitigate network jitter
- **Other** — Built an in-editor track builder for rapid greybox iteration, allowing designers to place a car and drive any section before handing the track to the art team



Motorplanet

Aug 2014
Sep 2014
2 mos + advisory support

8D STUDIO

System Architect & Lead Programmer

Joined for a predefined two-month Unity engagement on Sketch Tales, a single-player voxel adventure for Steam

- **Architecture** — Designed and implemented the game's core architecture
- **Render & TechArt** — Built the part-based character-rendering system at the center of the game
- **Other** — Handed development to the internal team as planned and continued advising through most of the project's later production



Sketch Tales

Aug 2013
Jul 2014
1 yr

WEST-GAMES

System Architect & Lead Programmer

Led technical development of Seaker, a client/server hidden-object game built around 3D panoramas

- **Architecture** — Owned technical direction and system architecture
- **Render & TechArt** — Designed a unified shader and material approach that reduced the game's rendering to roughly seven batches
- **Network** — Built the game's client/server functionality using Unity Legacy Networking

---

Feb 2012
Jul 2013
1 yr 6 mos

GOGAMES

System Architect & Lead Programmer

Led system architecture and hands-on C# server development for World of Fishing, supporting development of the Flash client

- **Architecture** — Designed the architecture for modular ASP.NET server logic
- **Network** — Defined the client/server communication protocol
- Led the backend data migration from SQL to MongoDB and Redis, establishing a stack later reused at West-Games and D.A.W.Studio


World of Fishing

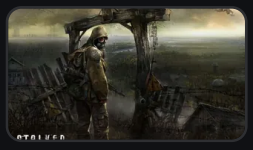
Dec 2009
Dec 2011
2 yrs 1 mo

GSC GAME WORLD

Tools & Game Logic Programmer

Developed game logic and production tooling for S.T.A.L.K.E.R. 2

- **Other** — Worked across C and C++ engine runtime and gameplay code, C# editor tooling, and the Managed C++ interop layer connecting them
- Built an editor-tooling framework including a property grid, curve editor, and foundational node editor
- Implemented gameplay systems including proximity triggers and centralized projectile management



S.T.A.L.K.E.R. 2

Jun 2007
Nov 2009
2 yrs 6 mos

SONELLY

Programmer

Developed game systems and engine infrastructure in C# for a custom XNA-based .NET 3D engine

- **Architecture** — Designed a JSON-based scene format supporting object hierarchies, components, loading, and persistence
- **Render & TechArt** — Developed a character-recoloring shader supporting multiple color variants
- Built scene-effect rendering and smooth transitions between scenes
- **Other** — Implemented player save and load persistence
- Developed hidden-object gameplay with an inventory system
- Implemented match-3 gameplay logic

ARCHITECTURE FOR FAST ITERATION AT SCALE

Let's build the foundation right

LinkedIn ↗

GitHub ↗

Jes Obertyukh

Unity · Rendering · Game Architecture