High-End Complex Algorithm Optimization Expert | UI & Runtime Engines, Interpreters — Advancing Toward High-Performance AI Graph & Vector Databases
하이엔드 컴플렉스 알고리즘 최적화 전문가 | UI & 런타임 엔진, 인터프리터 — 고성능 AI 그래프 & 벡터 데이터베이스를 향해 나아가다

1. Introduction / 소개

High-End Complex Algorithm Optimization Expert with 35+ years in C/C++ and 20+ years in R&D, specializing in pushing performance limits for UI engines, runtime interpreters, rendering pipelines, and large-scale data structures.
35년 이상 C/C++ 경력과 20년 이상의 R&D 경험을 바탕으로, UI 엔진·런타임 인터프리터·렌더링 파이프라인·대규모 데이터 구조의 성능 한계를 돌파해 온 전문가입니다.
I treat algorithms not as mere tools, but as living systems hiding untapped performance potential.
저는 알고리즘을 단순한 도구가 아닌, 숨겨진 성능 잠재력을 품은 살아있는 시스템으로 바라봅니다.

2. Key Achievements / 주요 성과

1) Global-Standard AutoLayout Algorithm: Reimagined & Rebuilt from Scratch

Reverse-engineered **Figma's world-renowned AutoLayout algorithm** purely through visual observation and behavioral inference — without access to its internal source.Developed a **pixel-perfect, runtime-compatible engine** supporting deeply nested WRAP/FILL/HUG constraints, rotated containers, and axis-aware alignment rules.Achieved full layout consistency across design and runtime by applying **enum-based dispatching**, **compile-time constant folding**, and **recursive edge condition flattening**.This algorithm, now powering ProtoPie Runtime, replicates the logic behind today's most influential UI design tool — not by mimicry, but through **original algorithmic intuition and structural mastery**.
피그마의 **세계적 AutoLayout 알고리즘**을 내부 구현 없이 오직 시각적 분석과 동작 패턴 유추만으로 완전 재현.중첩된 WRAP/FILL/HUG 조건, 회전 컨테이너, 축 기반 정렬을 모두 지원하며, 픽셀 단위의 정확성을 달성.분기 처리 최적화(enum dispatch), 상수화 기반 구조 단순화, 재귀 경계 안정화 기법으로 런타임 안정성과 성능을 모두 확보.현재 ProtoPie 엔진에 적용되어, 디자이너와 엔지니어 간 **레이아웃 신뢰의 경계를 허문 결정적 성과**로 작동 중.

2) Software Rasterization & Rendering Optimization

Re-engineered the original scanline-based rasterization pipeline (developed by a KAIST Ph.D. team) for Adobe ActionScript3, achieving over 5× performance gain via memory locality optimization, low-level algorithm tuning, and cache-friendly data structures.
KAIST 박사팀이 개발한 Adobe ActionScript3 스캔라인 기반 래스터라이저를 재구성하여, 메모리 로컬리티 최적화·저수준 알고리즘 튜닝·캐시 친화적 자료구조를 통해 5배 이상 성능 향상.

3) GPU Shader Acceleration & Parallel Processing

Reimplemented all Flash filter effects and blend modes as parallelized GPU shaders, leveraging SIMD/SIMT architecture, compile-time constant folding, and GPU kernel optimization.
Flash의 모든 필터와 블렌드 모드를 GPU 병렬 셰이더로 재구현하고, SIMD/SIMT 아키텍처·컴파일타임 상수 폴딩·GPU 커널 최적화를 적용.

4) Beyond-AOT Bytecode-to-Native Compilation

Converted ActionScript3 bytecode into highly optimized C++ native code, compiled into dynamic libraries; outperforming both JIT and AOT via aggressive compiler optimizations and IR tuning.

ActionScript3 바이트코드를 고성능 C++ 네이티브 코드로 변환·동적 라이브러리화하고, 공격적인 컴파일러 최적화 및 IR 튜닝을 통해 JIT과 AOT 모두를 능가하는 성능 달성.

5) Interpreter & JIT/AOT Hybrid Execution Engines
Designed high-performance interpreters and hybrid execution engines for DSLs such as ProtoPie Formula and KScript, with profiling-guided optimization and branch-prediction-aware execution.
ProtoPie Formula·KScript 등 DSL용 고성능 인터프리터 및 하이브리드 실행 엔진 설계, 프로파일링 기반 최적화 및 분기예측 친화적 실행 구조 구현.

6) Middleware & Cross-Platform Engine Architecture
Engineered cross-platform middleware frameworks with deep focus on caching strategies, branchless algorithms, and memory bandwidth optimization for real-time interactive systems.
실시간 인터랙티브 시스템에 맞춰, 캐싱 전략·브랜치리스 알고리즘·메모리 대역폭 최적화에 중점을 둔 크로스플랫폼 미들웨어 프레임워크 설계.

7) ProtoPie–Unreal Engine HMI Integration
Developed a custom Unreal Engine plugin for Hyundai Motor Company's HMI prototype, enabling multi-layer UI event passthrough, script binding, and complex UX orchestration.
현대자동차 HMI 프로토타입을 위해, 멀티레이어 UI 이벤트 패스스루·스크립트 바인딩·복합 UX 오케스트레이션을 구현한 언리얼 엔진 플러그인 개발.

8) WkBitwiseSortSet
Invented a branchless, cache-optimized binary tree traversal and branchless sorting algorithm via index reversal, enabling O(log N) set operations and scalability to millions of nodes.
인덱스 리버설 기반 브랜치리스·캐시 최적화 이진 트리 순회 및 분기 없는 정렬 알고리즘을 발명, O(log N) 집합 연산과 수백만 노드 확장성 달성.

9) Naver Entry Execution Mode Optimization
Optimized the execution mode of Naver Entry (originally developed by a KAIST master's-level team) for a 10× mobile performance boost, demonstrated in live prototypes.
KAIST 석사팀이 개발한 네이버 엔트리 실행 모드를 최적화해 모바일 성능을 10배 향상, 라이브 프로토타입으로 시연.

10) Unreal Engine 3 64-bit Transition
Successfully upgraded a full UE3 + Scaleform + third-party integration from 32-bit to 64-bit, ensuring stability across the entire toolchain.
UE3 + Scaleform + 서드파티 통합 프로젝트를 32비트에서 64비트로 성공적으로 전환, 전체 툴체인의 안정성을 확보.

3. Core Competencies / 핵심 역량

- **Performance-first Design** — Mastery of CPU/GPU pipelines, caching, branch prediction, memory models.
  성능 우선 설계 — CPU/GPU 파이프라인·캐싱·분기 예측·메모리 모델에 대한 깊은 이해.
- **Algorithm Versatility** — From rasterization to graph indexing, adaptable to any bottleneck.
  알고리즘 범용성 — 래스터라이징부터 그래프 인덱싱까지, 모든 병목에 대응 가능.
- **Low-level Optimization** — C++, assembly, compiler internals, branchless patterns.
  저수준 최적화 — C++, 어셈블리, 컴파일러 내부 구조, 브랜치리스 패턴.
- **Scalable Architecture** — Handles millions of nodes, real-time rendering, large-scale DB queries.
  확장성 있는 구조 — 수백만 노드·실시간 렌더링·대규모 DB 쿼리 처리 가능.

4. Future Direction / 향후 방향

Currently applying my optimization expertise to AI-focused Graph and Vector databases, aiming to push indexing and query performance beyond current limits.
현재 AI 중심의 그래프·벡터 데이터베이스에 최적화 전문성을 적용하여, 인덱싱 및 쿼리 성능을 현존 한계 이상으로 끌어올리는 연구 진행 중.

5. Practical Application Value / 실전 적용 가치

Every listed achievement is not just a theoretical experiment but has been battle-tested in real-world, high-stakes projects — from commercial game engines to enterprise-scale UI runtimes.
여기에 나열된 모든 성과는 단순한 이론적 실험이 아니라, 상용 게임 엔진부터 엔터프라이즈급 UI 런타임까지 실제 현장에서 검증된 고난도 프로젝트 경험입니다.
These optimizations have delivered measurable gains — often exceeding 5× performance — while maintaining full feature parity, stability, and scalability.
이러한 최적화는 기능·안정성·확장성을 그대로 유지하면서, 실제로 5배 이상의 성능 향상을 달성한 사례가 많습니다.
I specialize in extracting hidden performance from already high-end algorithms — the kind of work that most developers might encounter only once or twice in their careers.
저는 대부분의 개발자가 평생 한두 번 경험할까 말까 한 고급 알고리즘에서, 숨겨진 성능을 끌어내는 데 특화되어 있습니다.
If your system already feels "fast" but you suspect it could be faster, that’s when you call me.
당신의 시스템이 이미 “빠르다”고 느껴지지만, 더 빨라질 수 있다고 생각된다면 바로 그때가 저를 부를 때입니다.