GSS & AGSS Implementation TODO

🎯 Mission: Transform LUASCRIPT into the most elegant, powerful, and AI-assisted programming language

Milestone A: GSS Implementation (Elegance)

Phase 1: Grammar & Parser ⏳

- [ ] Define GSS grammar in LPEG
- [ ] Core grammar (Stylesheet, Block, Stmt)
- [ ] Field expressions (gaussian, mix, sum)
- [ ] Ramp, Iso, Blend statements
- [ ] CSS variable support (var(–name, fallback))
- [ ] Implement parser
- [ ] Lexer with token generation
- [ ] Recursive descent parser
- [ ] AST node construction
- [ ] Error recovery and reporting
- [ ] Semantic analysis
- [ ] Variable resolution
- [ ] Type checking
- [ ] Default value application
- [ ] Range validation
- [ ] Unit tests
- [ ] Parser correctness tests
- [ ] Error handling tests
- [ ] Edge case coverage

Phase 2: Kernel Graph IR ⏳

- [ ] Define IR node types
- [ ] GaussianNode
- [ ] MixNode, SumNode
- [ ] RampNode, IsoNode
- [ ] CompositeNode
- [ ] Graph construction
- [ ] AST → IR lowering
- [ ] Node dependency tracking
- [ ] Topological sorting
- [ ] CSS Variable Manifest
- [ ] Extract all CSS vars
- [ ] Generate defaults and ranges

- [ ] Parameter binding system
- [ ] Optimization passes
- [ ] Constant folding
- [ ] Dead code elimination
- [ ] Common subexpression elimination

Phase 3: Runtime Kernels ⏳

- [ ] Gaussian kernel
- [ ] Tile-based renderer (128×128)
- [ ] Precompute inv2s2
- [ ] Efficient exp() computation
- [ ] Batch processing (4-8 tiles)
- [ ] Ramp LUT system
- [ ] Built-in palettes (viridis, plasma, magma, inferno)
- [ ] Custom color stop interpolation
- [ ] 256-entry LUT precomputation
- [ ] Bilinear interpolation
- [ ] Marching squares iso
- [ ] Downsampled field (factor 2-4)
- [ ] 16-case lookup table
- [ ] Edge interpolation
- [ ] 1px stroke rendering
- [ ] Blend modes
- [ ] Normal, multiply, screen
- [ ] Softlight, overlay
- [ ] Alpha compositing
- [ ] Tile cache
- [ ] Cache key: (muX, muY, sigma, viewport)
- [ ] LRU eviction policy
- [ ] Separate caches for iso/ramp

Phase 4: Performance Optimization ⏳

- [ ] Benchmark harness
- [ ] FPS measurement
- [ ] Latency tracking (p50, p95, p99)
- [ ] CPU% and memory usage
- [ ] CSV output format
- [ ] Profile and optimize
- [ ] Identify bottlenecks
- [ ] Optimize hot paths
- [ ] Reduce allocations
- [ ] Vectorization opportunities
- [ ] WASM backend
- [ ] LuaJIT → WASM compilation
- [ ] WASI integration

- [ ] Uint8ClampedArray interface
- [ ] OffscreenCanvas + Worker
- [ ] Meet performance targets
- [ ] ≥60 FPS at 640×480 (single blob)
- [ ] ≥30 FPS at 1280×720 (two blobs + iso)
- [ ] First paint ≤300 ms
- [ ] Parameter update ≤60 ms

Milestone B: AGSS Implementation (Agentic Optimization)

Phase 5: Agent Grammar & Parser ⏳

- [ ] Define AGSS grammar extension
- [ ] @agent blocks
- [ ] optimize { } statements
- [ ] target, vary, budget, strategy, record
- [ ] Extend parser
- [ ] Parse @agent blocks
- [ ] Build agent AST nodes
- [ ] Validate agent semantics
- [ ] Parameter range specification
- [ ] ∈ [min, max] syntax
- [ ] step size support
- [ ] Multiple parameter ranges

Phase 6: Agent Runtime ⏳

- [ ] Search strategies
- [ ] Grid search
- [ ] Random search
- [ ] Bayesian optimization (Gaussian Process)
- [ ] Simulated annealing
- [ ] Performance measurement
- [ ] FPS tracking
- [ ] Latency measurement
- [ ] SSIM quality metric
- [ ] Custom metrics support
- [ ] Agent loop
- [ ] Parameter iteration
- [ ] Render and measure
- [ ] Reward computation
- [ ] Best parameter tracking
- [ ] Safety mechanisms
- [ ] SSIM threshold stop
- [ ] Timeout handling
- [ ] Resource limits

Phase 7: Tape-Deck UI Controls ⏳

- [ ] Control interface
- [ ] Play button (start/resume)
- [ ] Pause button (yield after tile batch)
- [ ] Stop button (cancel and flush)
- [ ] Repeat button (rerun best case)
- [ ] Record button (toggle logging)
- [ ] State management
- [ ] Running, paused, stopped states
- [ ] Progress tracking
- [ ] Best result caching
- [ ] Visual feedback
- [ ] Progress bar
- [ ] Current parameters display
- [ ] Real-time metrics chart

Phase 8: Logging & Reporting ⏳

- [ ] Structured logging
- [ ] Trial number
- [ ] Parameters (sigma, muX, muY, etc.)
- [ ] Metrics (fps, latency, SSIM)
- [ ] Checksum (for reproducibility)
- [ ] Timestamp
- [ ] Output formats
- [ ] CSV for data analysis
- [ ] Markdown for reports
- [ ] JSON for programmatic access
- [ ] Visualization
- [ ] Parameter sweep heatmaps
- [ ] Convergence plots
- [ ] Pareto frontier (multi-objective)

Integration & Testing

Phase 9: Full Integration ⏳

- [ ] IDE integration
- [ ] GSS editor with syntax highlighting
- [ ] Live preview canvas
- [ ] Parameter sliders
- [ ] Agent control panel
- [ ] End-to-end tests
- [ ] Parse → IR → Runtime pipeline
- [ ] Agent optimization loop
- [ ] UI control interactions

- [ ] Documentation
- [ ] User guide
- [ ] API reference
- [ ] Tutorial examples
- [ ] Demo gallery

Phase 10: Validation & Polish ⏳

- [ ] Acceptance criteria verification
- [ ] A1: Engine boundary + JS fallback
- [ ] A2: Benchmark harness with CSV
- [ ] A3: Baseline comparisons (SSIM)
- [ ] A4: GSS parse/compile (≤1 frame)
- [ ] A5: Agent loop (≥10 iter improvement)
- [ ] A6: WASM path passes tests
- [ ] Mathematical elegance
- [ ] Unicode operator support (∑, √, ∫, ∂, etc.)
- [ ] ASCII equivalence (≤1 ULP difference)
- [ ] Formula-to-pixels correctness proof
- [ ] Performance validation
- [ ] All targets met
- [ ] Benchmark results documented
- [ ] Comparison with baselines
- [ ] Code review
- [ ] Round-robin team reviews
- [ ] Steve Jobs design critique
- [ ] Donald Knuth correctness audit

Current Status

Branch: feat/gss-agss-implementation
Phase: Starting Phase 1 - Grammar & Parser
Progress: 0% → Target: 100%

Next Actions

1. ✅ Create design document (DESIGN_GSS.md)
2. ✅ Create TODO checklist (this file)
3. ⏳ Implement GSS grammar in LPEG
4. ⏳ Build parser with AST generation
5. ⏳ Add semantic analysis
6. ⏳ Write unit tests

Team Coordination

Grammar/Parser Group: Focus on Phases 1, 5
Core Engine Group: Focus on Phases 2, 3

UI/Tooling Group: Focus on Phases 7, 9
Testing Group: Focus on Phases 4, 8, 10

Round-Robin Reviews: Each group reviews another's work cyclically

Leadership Checkpoints

Steve Jobs: "Keep it minimal. CSS-like elegance. Ship fast."
Donald Knuth: "Prove correctness. Document everything. No shortcuts."

Last Updated: 2025-09-30
Next Review: Daily standup
Target Completion: 8 days