

GUTHRIE B ADAMS III

Senior / Principal Software Engineering Leadership

Austin, Texas | 512-762-9552 | guthrieadams@gmail.com
beastworks.com | github.com/gadams3 | linkedin.com/in/guthrieadams

PROFILE

I build the engines, systems, tools, and workflows that turn ambitious ideas into games, creative experiences, and shipped products. Across three decades of professional and independent development, I have shipped game titles across PC, console, and mobile platforms and applied that experience across industries. I combine hands-on software engineering with technical and team leadership, connecting architecture, animation, rendering, visual effects, simulation, and content creation.

I led major Lumberyard and O3DE initiatives spanning procedural world building, vegetation, terrain foundations, editor modernization, material and shader authoring, and rendering integration. Throughout my career, I have connected disciplines, mentored engineers, and helped teams carry systems from design through implementation, production use, and customer adoption.

My recent work brings that same technical depth and curiosity to navigation, geospatial services, and applied AI.

EXPERIENCE

Amazon

Senior Software Development Engineer | Applied AI Solutions | AWS Geospatial

Mar 2025 - Present

- **Turn-by-turn Navigation SDK and Starter App:** I owned the research, design, documentation, and development of a customizable navigation SDK and starter application for Amazon Location Service customers on iOS and Android. I built the initial implementation in Swift and SwiftUI, using abstraction layers to support MapLibre and Ferrostar back ends. I developed route conversion and editing systems and a custom interface with voice guidance, multi-stop planning, saved routes, and GPS or simulated movement.
- **AI-Assisted Development:** I applied AI-assisted workflows to SDK development, dependency research, documentation, and design analysis. I created interactive training exercises, analyzed survey data, generated diagrams, converted documents, and automated manual processes.
- **Operational Improvement Strategy:** I coordinated incident response and escalation work, then used operational experience and team feedback to propose improvements to training, documentation, developer setup, and automation. Teams adopted recommendations to reduce manual work and improve operational workflows.
- **Overture Maps:** I contribute to the Overture Maps open-source project as part of a focused effort on behalf of my organization.

Senior Software Development Engineer | Amazon Game Studios / AWS Game Tech

Oct 2016 - Mar 2025

- **Umbra Research and Engine Integration:** I led research and development into alternative uses of CPU occlusion culling beyond rendering, developing proposals and prototypes to evaluate those applications. I also led Umbra integration for O3DE, including an abstraction layer for alternate occlusion-culling implementations, geometry extraction, asset-builder integration, and sample-project optimization. I supported Unreal Engine plug-in integration and remote data-baking workflows with Deadline Cloud.
- **Application Security Guardian:** I promoted secure development practices, provided engineering guidance, and collaborated with security specialists throughout the software development life cycle.
- **Atom Leadership and Engine Integration:** I led the team responsible for integrating Atom's rendering capabilities with the engine, editor, and production workflows. I set priorities, delegated work, reviewed designs and code, and implemented systems across team boundaries. I worked with rendering, UX, technical art, physics, animation, tools, and core-engine teams to connect authoring with runtime behavior. I designed component adapter patterns that reduced boilerplate and standardized reflection, scripting, automation, and testing, then worked with principal engineers to establish those patterns in the core framework.
- **Material Canvas:** I initiated and led Material Canvas, writing the RFC and designing its graph-based material and shader authoring architecture. I developed a JSON-driven node and template system that translated graphs into material types, materials, and shader source files. Users could extend the node library through data rather than editor-code changes. I extended Graph Canvas and Graph Model for standalone authoring and connected source generation, asset processing, and live previews so users could build and evaluate materials interactively.

- **Material Authoring and Live Iteration:** I led Material Editor's evolution into a document-oriented authoring tool and developed Material Component's in-level assignment and override workflows. I connected dynamic material slots, unique instances, reflected properties, and previews to the asset pipeline. I also resolved builder, dependency, and instance-management issues that affected hot reloading, enabling changes from authoring tools to propagate into editor scenes without restarting.
- **Atom Tools Framework and Shared Infrastructure:** I created Atom Tools Framework by extracting reusable document, viewport, property-editing, asset-browser, scripting, and automation systems from Material Editor. The framework supported multiple documents and views, undo and redo, persistent layouts, and extensible document types. It provided the foundation for Shader Management Console, Material Canvas, and additional prototypes, reducing repeated implementation work. I also built shared preview rendering for models, materials, lighting presets, and material overrides.
- **Dynamic Vegetation and Procedural World Building:** I led Dynamic Vegetation for New World and Lumberyard, owning the core design and most of the implementation while working with another engineer and a technical artist. I transformed an early prototype into a modular system for live, non-destructive world authoring. Shape-defined areas, selectors, filters, and modifiers gave artists control over placement and variation directly in the editor. I separated multithreaded sector evaluation from queued instance creation and removal, integrating dynamic placement with the renderer's main-thread requirements.
- **Gradient Signal, Surface Data, and Terrain Foundations:** I designed and implemented Gradient Signal to combine procedural noise, images, position, slope, and other inputs into reusable placement controls. I built Surface Data so vegetation could use supported geometry beyond terrain, including meshes, shapes, roads, rivers, and water. I created dependency monitoring that updated affected areas when users changed entities, assets, or configuration. These systems enabled interactive authoring and formed architectural foundations for the terrain effort I initially led before transferring to rendering.
- **Editor Modernization:** I redesigned the Entity Inspector and Entity Outliner with UX, replacing a monolithic property tree with component cards and improving multi-entity editing. I implemented dependency-aware guidance, component reordering, enable and disable controls, and contextual add-component workflows. I built ordering and caching support for predictable hierarchy editing, including drag-and-drop reparenting and reordering. These changes made entity composition easier to inspect and manage; I contributed as a named inventor to the related patent.
- **Applied AI Exploration:** I pursued ML and AI training and independent study of generative AI, contributed to technical education, and collaborated on experiments with small language models for real-time inference on game client hardware.

My major early editor, vegetation, and terrain contributions began in Lumberyard before the transition to GitHub. My public O3DE contributions under gadams3 document the later portion of this work.

Maximum Play, Inc.

Senior Software Engineer

Apr 2015 - Sep 2016

- I led core engine and editor development for MaxPlay's collaborative game-development platform. I built the task-based entity and component architecture, reflection and serialization systems, and rendering components in C++. The runtime organized independent work across CPU cores, providing the foundation for multicore simulation and rendering.
 - I owned the editor viewport, camera controls, live editing, and scene reconstruction. I developed WebSocket and JSON command processing that reconstructed assets, object hierarchies, and components as clients built and updated scenes during collaborative editing sessions.
 - I led the flocking and Origami Sky VR demonstrations, using spatial hashing and multicore task execution to scale animated agents. I also contributed to hiring, performance optimization, and code reviews.
-

Multimedia Games, Inc.

Senior / Lead Software Engineer

Jul 2010 - Feb 2015

- I led three to five engineers, mentored colleagues across the engineering organization, and established code-review and implementation standards. I interviewed candidates, taught HLSL shader-development classes, and shared technical findings from industry conferences.
 - I developed casino titles and their gameplay, animation, visual effects, and presentation systems, collaborating with artists, designers, audio specialists, and mathematicians.
 - I built an After Effects-based skeletal animation workflow that delivered complex sequences within memory constraints and reduced manual content-placement work. I adapted an emerging animation workflow from another engine to meet The Kraken's shipping requirements, then reused the technology on subsequent titles. I developed the systems primarily in C++, with JavaScript, Java, and C# as needed.
-

Flying Wisdom Studios / Social Gaming Network

Lead Software Engineer

Aug 2009 - Jun 2010

- **Exo Planet:** I led four engineers on a multiplayer 3D action game for early iPhone hardware and the first iPad launch. I developed the core engine, rendering, gameplay, collision, animation, camera, and asset pipeline in C++, Objective-C, and GLSL.

- I implemented movement and camera behavior for traversing magnetic surfaces, optimized runtime performance, and extended the renderer from OpenGL ES 1.1 to OpenGL ES 2.0. I collaborated with teams in San Francisco and Buenos Aires on multiplayer, interface, and platform integration.
-

Super Happy Fun Fun, Inc.

Lead Software Engineer

Oct 2008 - Apr 2009

- **iOS Engine:** I ported the studio's 2D mobile framework to iOS and extended it to support real-time 3D games. I built the C++ and Objective-C runtime systems and a Maya export pipeline for game-ready levels and assets. This foundation supported both new titles and ports of existing games.
 - **Snowboarding TnT and Big Buck Hunter Pro:** I owned and developed gameplay, control, collision, effects, and object interactions for Snowboarding TnT, then extended the engine and rendering foundation for Big Buck Hunter Pro with richer 3D environments, animated characters, stronger lighting, and better handling of transparency and UI.
 - I also conducted regular candidate interviews.
-

NCSOFT

Senior Software Engineer

Nov 2005 - Aug 2008

- **Tabula Rasa:** I developed and maintained production tools for artists and designers on the PC MMO. I improved particle and animation-VFX workflows, terrain painting, map generation, model viewing, world editing, and 3ds Max integration. I also implemented runtime interface features and optimized related systems, working in C++, Python, and MEL.
 - **Dungeon Runners:** I led conversion of that team's particle system and related tools for PS3.
-

Multimedia Games, Inc.

Game Programmer

Sep 2004 - Nov 2005

- I ported casino games to new platforms and regions, then developed new titles with mini-games and touch-screen interaction. I collaborated with illustrators, animators, and other production specialists to deliver the games in C++ on Windows.
-

Iguana Entertainment / Acclaim Studios Austin

Game Programmer / Lead Programmer

Jun 1999 - Aug 2004

- **All-Star Baseball 2006:** I led programming and supported technical planning, profiling, optimization, and downloadable-content updates for the PlayStation 2 and Xbox project. The studio closed before the game shipped.
- **The Red Star:** I developed camera controls and prototyped side-scrolling aerial combat sequences in C++ for PlayStation 2 and Xbox. I also researched AI middleware for the studio's next-generation engine work.
- **NBA Jam 2004:** I improved locomotion and animation-state management, then independently designed and implemented the particle system, visual-effects editor, and runtime integration. I worked directly with artists to deliver authoring tools and real-time previews for PlayStation 2, GameCube, and Xbox production.
- **NFL Quarterback Club 2002 / 2003:** I led gameplay, AI, player-control, and animation-sequence work during a substantial migration to new C++ systems. I coordinated integration with evolving engine and content pipelines for PlayStation 2, GameCube, and Xbox. The 2003 project did not release following licensing changes.
- **Turok 3: Shadow of Oblivion:** I implemented three boss encounters in C for Nintendo 64, coordinating AI, attacks, animation, and arena behavior.
- **South Park Chef's Luv Shack:** I developed interface systems, score screens, mini-games, and animation behavior in C for Nintendo 64, PlayStation, PC, and Dreamcast.

EDUCATION

I completed three years of Electrical Engineering and Computer Engineering coursework at the University of Texas at Austin from 1996 to 1999, with additional study in Computer Science.

INDEPENDENT PROJECTS AND STUDY

I independently designed and developed BeastWorks CAGE from 2005 through 2015. I built the cross-platform engine, editor, and tools end to end, including rendering, reflection, messaging, asset processing, scripting, visual effects, live editing, and undo and redo. The technology supported three apps I shipped on the App Store.

CAGE served as a practical test bed for architecture, performance, and authoring workflows. I carried those lessons into later professional projects.