

RUI TANG (SCOTT)

ruitang591@gmail.com | 747-222-0001 | scott-tang.com | github.com/ttocs778 | linkedin.com/in/ttocs778

EDUCATION

University of Pennsylvania, School of Engineering and Applied Science

Master of Science in Engineering, Computer Graphics and Game Technology (CGGT)

Aug 2026 - May 2028 (Expected)

Philadelphia, PA

Relevant Coursework: Interactive Computer Graphics, Computer Animation, 3D Computer Modeling

University of Southern California, Viterbi School of Engineering

Bachelor of Science in Computer Science (Games), GPA: 3.80/4.00

Aug 2022 - May 2026

Los Angeles, CA

Relevant Coursework: Operating Systems, Distributed Systems, Game Engine Development, Game Design & Production

SKILLS

Languages: C++, C#, Python, Java, HLSL, Lua, SQL

Graphics & Rendering: DirectX 11, OpenGL, HLSL, Shader Programming, Skeletal Animation, Post-Processing Pipelines

Engines & Tools: Unreal Engine 5, Unity, SDL2, Perforce, Git, Visual Studio

Systems: Multi-threading, Networking & Replication, Memory Management, SIMD, Profiling & Optimization

EXPERIENCES

ByteDance | Douyin - TikTok Parent Company

May 2025 - Aug 2025

Software Engineer Intern - 3D Engine Development

Beijing, China

- Shipped interactive social features (co-op pet raising) for Douyin reaching **10M+ DAU**; prototyped AIGC-driven UGC gameplay systems.
- Architected a modular asset pipeline in Unity, integrating with ByteDance's SAR engine, using a factory method pattern with staged processing (pre/post workflows). Built extensible modules and custom Unity Editor tools to transform meshes, shaders, and materials into deployable SAR bundles through a TypeScript interface toolchain, enabling cross-engine publishing and reducing manual conversion overhead.
- Implemented a custom Dynamic Bone runtime system mapping JSON-defined colliders to skeleton nodes and constructing recursive particle chains; optimized simulation stability and performance, fully replacing the previous solution across platforms.
- Optimized client performance for group chat and low-end devices through LOD replacement and frame cache mechanisms; resolved 20+ shader parameter inconsistencies between Unity and SAR to improve rendering stability and visual consistency.

NetEase Games | One of China's largest game developers

May 2024 - Aug 2024

Game Designer Intern

Hangzhou, China

- Designed and iterated over 10+ interactive levels for *Eggy Party* using NeoX engine's internal editor and scripting tools, focusing on obstacle placement, level progression, difficulty tuning, and UGC template design to drive player engagement across varying skill levels.
- Collaborated with design leads, engineers, and artists to refine gameplay mechanics and flow; led structured playtests and adjusted level layouts and parameters to improve onboarding experience, game balance, and win-rate consistency.
- Analyzed heatmaps and player behavior data via internal dashboards to identify failure hotspots and early churn; refined trap timing and reward pacing, contributing to an **~8% increase in Day 3 retention** during live seasonal content rollout.

iFlytek | Leading AI company in China

May 2023 - Aug 2023

Software Engineer Intern

Shanghai, China

- Integrated real-time ASR and NLP models into iFlytek's intelligent simultaneous interpretation system by implementing model invocation interfaces and aligning time-stamped text streams with audio input in C++.
- Utilized Valgrind to trace and fix memory management issues in the live captioning module, improving system stability for long translation sessions.
- Optimized the ASR decoding pipeline and downstream NLP integration, reducing average processing delay by **~150ms** in test environments.

SELECTED PROJECTS

Lucky Duckies (Unreal Engine Co-op Game) | [Project Page](#) | [Steam](#)

Aug 2025 - Jun 2026

Systems & UI Engineer

- Engineered gameplay systems and fixed production issues in a 100,000+ LOC Unreal Engine codebase within a 40+ member cross-disciplinary team using sprint-based development.
- Developed the dialogue framework on Unreal's replication architecture, implementing dialogue state management, server-authoritative logic, and RPC synchronization to ensure consistent NPC interactions across clients.
- Implemented HUD and world-space UI (tutorials, progression, boss health, prompts), ensuring responsive updates under a networked environment.
- Engineered immersive cutscene flow and a hybrid online/local character selection level for co-op split-screen experience, enabling multiplayer state transitions and seamless input hot-swapping (controller/mouse).

DirectX11 Custom Engine | [Project Page](#) | [GitHub](#)

Jan 2025 - May 2025

Engine Programmer

- Built a C++ game engine on DirectX 11 from an empty Win32 window, with a component-based scene loaded from JSON levels, a templated asset cache, and an SSE (SIMD) math library backed by ~1,450 lines of unit tests.
- Implemented a rendering pipeline with Phong lighting (up to 8 point lights), tangent-space normal mapping, and a multi-pass bloom post-process using off-screen render targets, half/quarter-resolution downsampling, and a separable 9-tap Gaussian blur.
- Developed skeletal animation with keyframe interpolation and GPU skinning (4 bones per vertex), layered FABRIK inverse kinematics on top via a pose callback, and offloaded per-character animation to a 4-thread job system profiled with a custom scoped timer exporting Chrome trace data.

Slash Bash (Unity Action Fighting Game) | [Project Page](#) | [itch.io](#)

Jul 2024 - Jan 2025

Lead Gameplay Programmer & Combat Designer

- Built a data-driven combat system: all 36 moves across 9 characters are ScriptableObject configs whose hitboxes (circle or sector, via `OverlapCircleAll`) spawn on a set animation frame, so balance passes needed no code changes, and the whole game uses a single hitbox prefab.
- Implemented local 2-player controller input with Unity's Input System, turning button and stick presses into symbols matched against each skill's combo string (e.g. B B A) with a 0.5s timing window, covered by NUnit tests.
- Built per-character state machines that load sprite sheets by state name (no Animator), arena hazards (burn-over-time traps, raycast-blocked obstacles), and a debug scene previewing all 9 characters' animations at once.
- Published a demo on itch.io with 100+ downloads in the first week and a 4.5/5 average rating based on player feedback on animations and variety.

Classic Game Remakes | [Project Page](#)

Aug 2024 - Dec 2024

Engine & Gameplay Programmer

- Built an actor-component game engine in C++ with SDL2 and an OpenGL 3D renderer, used across 9 games with delta-time updates, AABB collision resolution, A* pathfinding, ghost AI, and an AI kart driving on the player's own physics; ported all 9 to the browser via Emscripten.
- Recreated *Portal*'s core mechanics around a single portal transform (inverse world matrix, 180° flip, exit matrix) reused for player teleport with momentum, the portal camera, lasers and energy pellets, with quaternion-based portal placement, a 6-state turret FSM and deterministic input replay.