Tristan Metz



Gameplay & Tools Programmer

I make characters feel good to control, and build the tools and test levels that let a team tune them fast. Lead programmer at XR Japan in Tokyo.

Basic info

Email
tristan.a.metz@gmail.com

LinkedIn
linkedin.com/in/tristan-metz

GitHub
github.com/123tris

itch.io
123tris.itch.io

Location
Tokyo, Japan

Nationality
Dutch (EU)

Education
Bachelor of Creative Media and Game Technologies
Saxion University of Applied Sciences, Enschede, the Netherlands

Skills

Gameplay
Character controllers and game feel, physics (including my own engine), VR interaction, local multiplayer, dialogue and event systems

Tools
Unity editor tooling, level editors, serialization and reflection, test levels and test environments

Engines
Unity (since 2016), custom C++ engines (Vulkan, OpenGL), Unreal Engine

Programming languages
C#, C++, HLSL, Lua

Middleware
FMOD, Burst/Jobs, Mirror, XR Interaction Toolkit, ImGui, Git

Languages
Japanese (JLPT N1), English (native level), Dutch (native)

Summary

Game programmer focused on how a game feels to play and on the tools behind it. My recent work is in C# and Unity; before that I wrote the physics, serialization and editor code of two custom C++ engines.

Work history

Lead Programmer

March 2025 — Present
XR Japan (formerly AVR Japan), Tokyo, Japan

- Lead programmer on Lumen, a virtual trainer in which doctors practise endoscopic submucosal dissection (ESD) with a dedicated endoscope controller and foot pedal, and on VR and iPadOS training apps for clients
- Tissue behaviour on Burst-compiled solvers, so the simulation stays smooth at a high frame rate; lesion marking; and the mapping from the controller and pedal to the endoscope tip (push and pull, rotation, angle)
- Performance measurement and frame-time work on the simulation, and faster loading of its mesh data
- Architecture, conventions and code reviews; mentoring and onboarding programmers; the technical side of conversations with doctors, partners and clients, including estimates and risk tracking
- Built the shared Unity packages used across projects (Ripple for events and state, a VR template, UnityUtilities, AutoAssign) and test environments that take repetitive checks off the team

Game Programmer

September 2024 — February 2025
AVR Japan, Tokyo, Japan

- Gameplay, UI and tools for VR and tablet training apps in Unity, with some Unreal Engine work
- Wrote the VR template package new client apps start from: interaction, fade, tween and conditional-event components, so each new client app starts from working parts instead of copied code

Co-founder & Programmer

January 2022 — Present
Idomi Games (indie studio, alongside full-time work)

- Programmer of Ryze, a 2D precision platformer with a Steam playtest, which began as our Global Game Jam 2022 entry; team of five
- The character controller around Ryze's one verb, throwing a pearl and teleporting to it: a small state machine where a throw pressed just before landing is buffered, walking off a ledge turns you into the pearl instead of dropping you, and probes under each foot let the animation react before you slip off an edge
- A test level with a station for every mechanic I built, from crumbling platforms to time bubbles that slow physics and animation together through per-object timelines (Chronos), so a controller change is checked against all of them in a minute
- A dialogue system carried over from an earlier project: it sends each line as an event and never touches the UI, and starting a dialogue is one awaitable call that takes the condition for advancing, so trigger zones, walk-up cutscenes and lines for repeated deaths are each a small component

- All programming on the studio's jam games, including Shroomer (GMTK Game Jam 2026, 4 days) and This Side Up (Mini Jam 188, 72 hours)

Gameplay Programmer (graduation internship)

September 2021 — April 2022

VR Amsterdam, Amsterdam, the Netherlands

- Sole programmer of Ghostebash, a multiplayer VR arcade game where groups share one real room, starting from an empty project
- Local multiplayer with Mirror on a client-host model, IK rigs that map each player's real body into the game, and physics-based grabbing and interaction with the XR Interaction Toolkit
- Graduation research on free-roaming VR: the network code cost 0.08 ms per frame with no clients, 0.40 ms with four and 1.33 ms with ten, which showed a client-host model was good enough

Programmer Intern

February 2019 — July 2019

Damocles Games, the Netherlands

- Turn-based strategy game on a 3D grid (Fire Emblem-like) in Unity, in a team of ten
- An API and custom units for a visual-scripting plugin, so designers could build gameplay behaviours without waiting for a programmer, which sped up playtest iteration
- A request for pawn hover highlighting showed how rigid the selection code was, so I rewrote the selection and highlight framework; after that, features like hovering and turn-based camera controls were quick to add

Engine and tools projects

Tristeon

C++ engine with a Vulkan renderer · 2 people, 5 months · 2017 · 70+ stars on GitHub

I wrote the editor in ImGui and the reflection behind it. C++ has no reflection, so every type registers a factory before main runs; scenes store type names, and the same data drives an inspector that edits any component without hand-written UI.

Orb

Puzzle game on our first C++ engine (OpenGL) · 2017

Wrote the physics engine: rigidbodies with drag, friction and bounciness, AABB collision with continuous collision detection so fast objects never tunnel through walls, triggers and raycasts. Made Unity the level editor with an exporter that the engine reads back object by object.

osu!catch level editor

Unity · solo · 2018 — 2019

A beat grid drawn by one fragment shader in pixel space, so lines stay exact at any tempo and zoom where thousands of line objects would be slow; undo and redo built from mementos that revert themselves; shaped by feedback from the game's mappers.

C# vs. Wren

Scripting benchmark for a C++ engine · 2023

Measured both before choosing one: C# ran 6 to 15 times faster, while Wren compiled over a thousand times faster and its calls into C++ were close to free.