MCP SERVER | NO CODE | CLOUD HOSTED

Matrix 4x4 Transforms MCP for AI Agents

Creating and manipulating 3D coordinate systems for graphics rendering

The Matrix 4×4 Transforms MCP handles all necessary linear algebra for advanced 3D graphics and simulation. It lets your AI client construct complex transformation pipelines by generating, composing, and applying specialized matrices—including translation, rotation, scale, and shear. You'll get precise control over model, view, and projection stages, converting between formats like quaternions and Euler angles to keep your rendering math accurate.

A+ Quality Score 100/100

matrix | transform | rotation | quaternion | euler-angles | homogeneous-coordinates

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JULY 2026
vinkius.com/mcp/matrix-4×4-transforms

SCAN TO VIEW

The connectivity layer between AI and the world's software.

Vinkius sits between AI and every application. All communication passes through Vinkius Cloud via the Model Context Protocol (MCP) — with governance, observability, and security at every layer.

Your AI Connections Run Through Vinkius Cloud

The world's largest managed MCP catalog

Vinkius is the connectivity layer where AI connects to the software your business already runs. We handle the hosting, the security, the credentials, the uptime — you get agents that actually do things.

We operate the world's largest managed MCP catalog. Major SaaS platforms, CRMs, databases, and cloud providers — running, monitored, production-ready. This MCP server is hosted and maintained by the Vinkius Cloud for AI Agents.

> *The agent doesn't manage credentials, doesn't manage uptime, doesn't manage security. Vinkius does.*
>
> — Architecture principle

Four Pillars of the Vinkius Runtime

01 — Security by design

Credentials stay encrypted at rest via AES-256. The AI agent never touches raw keys — they're injected into a sandboxed V8 isolate at runtime. Actions are logged, and connections have an emergency kill switch.

02 — Built on MCP Fusion

This MCP server was built with **MCP Fusion**, the open-source framework (Apache 2.0) that powers the entire Vinkius catalog. Schema-as-firewall strips undeclared fields, compiled PII redaction runs at zero overhead, and cryptographic lockfiles produce git-diffable audit trails.

03 — Deterministic observability

Eight immutable metrics per endpoint: request volume, p95 latency, error rate, active connections, cost attribution. A live payload feed logs every tool call with mutation detection.

04 — Autonomous operations

Servers are deployed, monitored, and patched autonomously. New capabilities and security patches ship weekly. Zero-downtime deployments ensure continuous availability across all managed MCP servers.

AES-256	Ed25519	24h TTL	V8 Isolate
Encryption at rest	PKI vault signatures	Ephemeral session keys	Sandboxed execution

One Token. Instant Access.

Every MCP server on Vinkius is accessed through a **Connection Token**. Tokens are generated in the cloud dashboard and produce a unique MCP endpoint URL. Paste this URL into any MCP-compatible client — no SDK required.

A single token can serve **multiple AI clients simultaneously**, or you can issue separate tokens per client for granular access control. Each token tracks its own request count, last activity timestamp, and can be individually enabled or revoked.

MCP ENDPOINT

```
https://edge.vinkius.com/{token}/mcp
```

Claude · Cursor · VS Code · Windsurf · Grok · Gemini

Security Is the Architecture

Security in Vinkius is not a feature — it's the foundation of the runtime. The gateway enforces multiple independent protection layers between AI agents and third-party APIs.

01 — Ed25519 PKI Vault

Every workspace has an Ed25519 Master Key. Session keys are generated ephemerally (24h TTL) and signed by the Master Key. Credentials never leave the vault boundary.

02 — V8 Isolate Sandboxing

Tool code runs inside isolated-vm V8 isolates with 64 MB memory caps and per-request timeouts. No filesystem access, no network access except through the SSRF-guarded fetch bridge.

03 — SSRF Guard

All outbound HTTP requests are DNS-resolved and validated before execution. Private IP ranges (10.x, 172.16-31.x, 192.168.x, AWS metadata 169.254.x) are blocked at the network layer.

04 — DLP & PII Redaction

A ResponseGuard pipeline intercepts every tool response. Configurable redaction patterns strip sensitive fields (emails, SSNs, card numbers) before data reaches the AI agent.

05 — Cryptographic Audit Trail

Every request is signed into a SHA-256 hash chain with Ed25519 signatures. Events form a tamper-proof, SIEM-exportable forensic record.

06 — Honeytoken Trap System

Phantom credentials are injected into isolated environments. If a honeytoken is used outside Vinkius infrastructure, the server is quarantined instantly.

Emergency Kill Switch

EU AI Act Art. 14(1) Compliant

The kill switch is an **emergency halt** mechanism — not a simple toggle. When triggered, it executes three actions atomically:

01 — Server deactivated

The MCP server is immediately taken offline across the entire cluster.

02 — All tokens revoked

Every connection token is invalidated. Total lockout — reconnection blocked until new tokens are issued.

03 — WebSocket connections killed

Active connections terminated via Redis pubsub broadcast. Propagates to every runtime node in the cluster.

Full Visibility. Zero Guesswork.

The Vinkius cloud dashboard includes a full MCP Governance suite — real-time analytics and security controls for production AI operations.

Control Plane

KPI dashboard with request volume, latency, success rate, token consumption, and AI-generated operational briefings.

Agent Activity

Which AI clients are connecting, how often, and what they're doing — real-time session tracking.

FinOps

Cost tracking per tool, payload compression savings, budget optimization signals, and consumption trends.

Tool Health

Slowest and most error-prone tools, with actionable root-cause insights and performance baselines.

Firewall & DLP

PII redaction activity, sensitive data protection counters, and security event timeline.

Incident Log

Error trends, failure rates, status-code breakdowns, and forensic audit trail access.

> Get started at **cloud.vinkius.com** — connect your AI agent in under 60 seconds.

Matrix 4x4 Transforms MCP

11 tools available
Cloud-hosted on Vinkius

Building a robust 3D graphics engine requires managing complex transformations, and this MCP gives you the tools for it. It lets your AI client generate foundational matrices—like those that position an object (translation) or orient it (rotation)—and then combine them in the correct mathematical order to create one composite matrix. You can also take a raw 3D point and apply a full transformation, managing perspective division automatically. For geometry pipelines, it's critical to switch between different representations, like converting rotation matrices into quaternions for smoother interpolation or extracting Euler angles when needed. If you're building simulations, this MCP provides the necessary structure; just connect it via Vinkius and your agent handles the heavy lifting of matrix math.

Core Capabilities

01 — Build a comprehensive transformation pipeline

Combine multiple specialized matrices into one final matrix that defines an object's complete position, orientation, and size in the scene.

02 — Define basic spatial transformations

Generate core 4×4 matrices for simple shifts (translation), uniform sizing changes (scale), or skewing/shearing along axes.

03 — Transform coordinates in 3D space

Apply a complete transformation matrix to an individual 3D point, handling perspective division correctly for accurate rendering.

04 — Convert rotational data formats

Switch between rotation matrices, quaternions, and axis-angle representations, solving common mathematical problems like gimbal lock detection.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/matrix-4×4-transforms** — connect your AI agent in three steps.

01 Start by generating the necessary foundational matrix (e.g., using `create_translation_matrix` or `create_rotation_matrix`) based on your desired shift, size, or orientation.

02 Use `compose_transforms` to multiply these matrices in the correct order, creating a single composite matrix that defines the object's final state for rendering.

03 Finally, pass this composite matrix and the target 3D point into `transform_point` to get the resulting coordinates after all transformations are applied.

The bottom line is you feed it raw geometric data—a point or a list of required transforms—and it returns the mathematically correct output for your graphics engine.

Built For

This MCP is essential for geometry and simulation developers. If you write code that needs to know where an object *should* appear in a virtual world, this is what you need. It's for people who get frustrated when their models look wrong because the transformation math was off by one multiplication.

Game Engine Programmer

Needs to calculate and apply complex camera or character movement matrices (view and projection) every frame, ensuring proper coordinate system mapping.

Visualization Engineer

Requires accurate model placement in scientific simulations, often needing to transform points from one local coordinate system into a global scene context.

Graphics Developer

Must build the core rendering pipeline logic, handling everything from object scaling and shearing to calculating final vertex positions using combined matrices.

What Changes When You Connect

01 Build complete object placements: Use `compose_transforms` to combine scale, rotation, and translation into one matrix, eliminating manual multiplication errors.

02 Handle data conversion instantly: Need to switch between formats? Convert rotations using `quaternion_to_rotation_matrix` or back with `matrix_to_euler_angles` without writing complex math code.

03 Perfect point placement: Never guess where an object ends up again. Use `transform_point` to apply any full matrix transformation to a specific 3D coordinate.

04 Define core geometry changes: Instantly generate base matrices for movement (`create_translation_matrix`), sizing (`create_scale_matrix`), or skewing (`create_shear_matrix`).

05 Robust rotation handling: Use `rotation_matrix_to_quaternion` to stabilize calculations and avoid issues like gimbal lock when interpolating object orientation.

Real-World Applications

Placing a character in the virtual world

The agent needs to move a character model from its starting point (A) to a target position (B). It uses `create_translation_matrix` and then combines that with existing rotation matrices to ensure the final placement is mathematically sound.

Simulating camera movement

The simulation needs to apply both translation (moving through space) and rotational view changes. The agent uses `compose_transforms` to build a combined View Matrix, giving the AI client precise control over the camera's perspective.

Exporting geometry for rendering

A model needs to be scaled up by 3x and then rotated 45 degrees before being viewed. The agent uses `create_scale_matrix` and `create_rotation_matrix`, composing them together to pass the final matrix to the renderer.

Analyzing object orientation data

The system receives a raw rotation matrix from an external source. The agent uses `matrix_to_euler_angles` to immediately understand the physical pitch, yaw, and roll of the object for debugging or display purposes.

Patterns to Avoid

Manually calculating combined matrices

✗ AVOID

Trying to manually multiply a scale matrix by an offset translation vector in code. This is slow, error-prone, and often misses the correct order of operations.

✓ INSTEAD

Use `compose_transforms`. You simply pass your foundational matrices (scale, translate) into this function, and it handles the necessary multiplication sequence for you.

Confusing point transformation vs. matrix composition

✗ AVOID

Applying a transform to a point *before* composing other transforms. This yields incorrect results because the point must be transformed by the final combined matrix.

✓ INSTEAD

First, use `compose_transforms` to build your single master matrix. Then, pass that complete matrix and the coordinate into `transform_point`. That's the right sequence.

Ignoring rotation format conversions

✗ AVOID

When interpolating between two object rotations, using raw Euler angles often results in unnatural 'snapping' or gimbal lock failures.

✓ INSTEAD

Always convert to quaternions first. Use `rotation_matrix_to_quaternion` and then let the agent handle the interpolation, ensuring smooth, stable rotation paths.

The Right Fit

Use this MCP if your core problem involves calculating or manipulating 4x4 matrices for graphics rendering. Specifically, if you need to combine shifts (translation), rotations, and sizes (scale) into a single, authoritative matrix—that's where this shines. Don't use it if you just need basic linear algebra; simple vector math is fine without the full matrix pipeline. You should also use it if your workflow requires switching between rotation formats like quaternions or Euler angles for different parts of the system (e.g., using quaternions for smooth animation but needing Euler angles for user input display). However, don't rely on this MCP to solve physics collision detection; while transformations are used in physics, actual force and impulse calculations require separate dedicated tools.

Matrix 4x4 Transforms MCP: Solving Complex 3D Graphics Pipelines

When building a graphics application, the manual process involves defining geometry transformations—scaling an object, moving it across the map, and then rotating it to face the player. Each step requires generating a specialized matrix (a translation matrix here, a rotation matrix there). Then, you have to manually multiply these matrices together in the exact order (e.g., Scale *Rotate* Translate) or your model will end up in the wrong place.

With this MCP, your agent handles that entire pipeline. You tell it what sequence of actions is needed—like scaling first, then rotating, and finally translating—and it generates the single, correct composite matrix for you. It takes away the risk of mathematical ordering errors.

Matrix 4x4 Transforms MCP: Managing Rotation Data in Simulations

In simulation development, rotation is tricky. If you rely solely on Euler angles for animation, you run

This MCP solves that complexity. It provides multiple conversion tools, letting you seamlessly

into gimbal lock—a failure state where two axes align and the system loses a degree of rotational freedom. You also have to manually convert between matrices and quaternions just to keep your data models consistent across different systems.

move from a rotation matrix to a quaternion for animation (avoiding lock) and back to Euler angles when the UI needs to display yaw/pitch/roll. You get stability and flexibility.

11 Tools Available

#	TOOL	DESCRIPTION
01	axis_angle_to_quaternion	Converts a rotation defined by an axis and angle into a quaternion format.
02	compose_transforms	Combines several separate transformation matrices into one single, usable composite matrix.
03	create_rotation_matrix	Generates a 4×4 rotation matrix around a specified axis for any given angle.
04	create_scale_matrix	Creates a 4×4 matrix that uniformly or non-uniformly scales objects along the X, Y, and Z axes.
05	create_shear_matrix	Generates a 4×4 shear matrix, allowing you to skew an object's geometry across its axes.
06	create_translation_matrix	Produces a 4×4 transformation matrix used specifically for shifting objects from one coordinate point to another.
07	matrix_to_euler_angles	Extracts the corresponding Euler angles (roll, pitch, yaw) from an existing rotation matrix.
08	quaternion_to_axis_angle	Converts a quaternion representation of rotation back into an axis-angle pair.
09	quaternion_to_rotation_matrix	Converts a quaternion value directly into a standard 3×3 rotation matrix.
10	rotation_matrix_to_quaternion	Takes an existing 3×3 rotation matrix and converts it into its quaternion equivalent.
11	transform_point	Applies a full 4×4 transformation to a single 3D point, adjusting for perspective division in the process.

See It in Action

Real prompts you can use once this MCP is connected to your AI agent through Vinkius Cloud.

U I need to compose a transform that moves my character 10 units forward, scales it up by 1.5x, and rotates it 30 degrees around the Y-axis.

The combined transformation matrix is ready. It incorporates scaling (1.5 on all axes), rotation (30° about Y), and translation (10 units Z).

Composite Matrix:

```
[ [ 1.5, 0, 0, 10 ]
[ 0, 1.5, 0, 0 ]
[ 0, 0, 1.5, 0 ]
[-0.5, 0, 0, 1 ]
]
```

Use this matrix to render the model; it correctly handles all three stages of transformation.

U Convert a rotation from a quaternion into an axis-angle format for logging purposes.

Quaternion Input: (0.5, 0.5, 0.5, 0.5)

Conversion Result:

- **Axis:** (1/√3, 1/√3, 1/√3) ≈ (0.577, 0.577, 0.577)
- **Angle:** 125.26°

The rotation is equivalent to an angle of approximately 125 degrees around the axis pointing through equal coordinates.

U Apply a full transformation matrix to the point (5, 10, 3) and tell me where it ends up.

Input Point: (X=5, Y=10, Z=3)
Applied Matrix: *[Full 4×4 Matrix]

The resulting transformed coordinates are approximately **(2.1, 8.9, 7.6)**.

Note: The final coordinate set is calculated after applying the perspective divide (W component).
This means the point has been correctly projected into the view space.

Frequently Asked Questions

01 **What is homogeneous coordinates and why are they used?**

Homogeneous coordinates represent 3D points as 4D vectors [x, y, z, w]. This representation enables translation and projection operations to be expressed as linear transformations using 4×4 matrices. Points are converted back to 3D by dividing by the w-component (perspective divide) when w ≠ 1.

02 **How do I compose multiple transformation matrices?**

Use the `compose_transforms` tool with an array of matrices in application order. The first listed matrix is applied first to the point. Composition uses right-to-left multiplication: if a point transforms first by matrix A then by matrix B, the composition is B × A.

03 **What is gimbal lock and how does this tool handle it?**

Gimbal lock occurs when rotation axes align, causing loss of a degree of freedom. The `matrix_to_euler_angles` tool detects this condition and returns a `gimbal_lock` flag with an explanation. This helps identify problematic rotation sequences in your graphics pipeline.

04 **Why use quaternions instead of rotation matrices?**

Quaternions avoid gimbal lock and are numerically stable for interpolation and composition. The `quaternion_to_rotation_matrix`, `quaternion_to_axis_angle`, and `axis_angle_to_quaternion` tools enable seamless conversion between quaternion and matrix representations for optimal performance in different scenarios.

Go Live in 60 Seconds

Get your connection token from **cloud.vinkius.com**, then paste the endpoint URL into any MCP-compatible client.

YOUR MCP ENDPOINT

```
https://edge.vinkius.com/[TOKEN]/mcp
```

CLIENT	WHERE TO CONFIGURE
Claude AI	Profile → Customize → Connectors → "+" → Add custom connector → Paste endpoint
Cursor	Settings → Features → MCP Servers → "+ Add New MCP Server" → Type: SSE → Paste endpoint
VS Code	Ctrl/Cmd+Shift+P → "MCP: Add Server" → add `"matrix-4x4-transforms": { "url": "…" }`
Windsurf	MCP Settings → `mcp_settings.json` → Add endpoint URL
ChatGPT	Settings → Tools & plugins → Add MCP server → Paste endpoint
Gemini	Extensions → Add MCP Server → Paste endpoint URL

ASK AN AI ABOUT THIS

Let your preferred AI explain this MCP server

- Ask ChatGPT
- Ask Claude
- Ask Perplexity
- Ask Gemini
- Ask Grok

READY TO CONNECT

Matrix 4x4 Transforms is live on Vinkius Cloud.

Get your connection token, paste it into your AI agent, and start building. No SDK. No deployment. Just results.

Start at cloud.vinkius.com →

vinkius.com · support@vinkius.com

INDEPENDENT PLATFORM DISCLAIMER

DOCUMENT INFORMATION

Generated	July 2026
MCP Server	Matrix 4×4 Transforms MCP
Server ID	`019f1e48-48c8-7286-b21a-367d7c43b0bc`
Platform	Vinkius Cloud for AI Agents
Endpoint	`https://edge.vinkius.com/{token}/mcp`

LICENSE & USAGE