MCP SERVER | NO CODE | CLOUD HOSTED

# Collision Detection Primitives MCP for AI Agents

## Calculating Precise 3D Intersections for Game Physics Simulations

Collision Detection Primitives is a specialized computational engine for 3D spatial mathematics. It calculates intersections, penetrations, and impact timings between common geometric shapes like spheres, axis-aligned bounding boxes (AABBs), rays, and planes. This MCP lets your AI agent determine exactly if objects overlap in virtual space or what point they hit along a path, giving developers precise physical data needed for realistic simulations.

A+ Quality Score 100/100

3d | collision | intersection | aabb | ray-casting | spatial-math

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JULY 2026
vinkius.com/mcp/collision-detection-primitives

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.

# Collision Detection Primitives MCP

4 tools available
Cloud-hosted on Vinkius

Building accurate 3D worlds requires knowing when things touch. Instead of writing complex math functions in C++ to check for overlaps, you let your AI client handle the heavy lifting. This MCP provides a computational layer that specializes in high-precision geometric intersection testing. You can ask if two spheres are overlapping and get the exact penetration depth. Need to see if a laser beam hits a virtual wall? The engine calculates the precise time of impact for rays against bounding boxes. For spatial containment, it verifies if an object is fully enclosed by specific boundaries. Connecting this MCP via Vinkius gives your agent access to industry-standard physics checks, letting you focus on design instead of differential geometry equations.

## Core Capabilities

**01 — Determine Sphere Overlap and Contact Details**

Checks if two spherical objects intersect and returns data like the depth of overlap and the precise contact point.

**02 — Test Collision Between Spheres and AABBs**

Calculates whether a sphere is colliding with an axis-aligned bounding box, crucial for character interaction checks.

**03 — Calculate Ray Impact Time Against Bounding Boxes**

Determines the exact time of impact and hit location when a simulated ray passes through or hits an AABB.

**04 — Verify Object Containment within Bounds**

Checks if a sphere is completely contained inside defined, specific spatial boundaries.

## One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/collision-detection-primitives** — connect your AI agent in three steps.

01 You give your AI client the parameters for two shapes (e.g., Sphere A and Sphere B) or a shape and its target boundary.

02 The MCP sends these geometric coordinates to the specialized computational engine, which runs multiple intersection algorithms in parallel.

03 Your agent receives a clean output: a simple 'yes' or 'no', along with critical metrics like penetration depth, contact points, or time of impact.

The bottom line is your AI client executes complex 3D physics calculations without you needing to write any geometry code.

## Built For

This MCP is essential for simulation engineers and game developers who deal with spatial constraints. If your work involves anything that moves in a virtual or physical space, these tools save hours of complex math coding by making intersection checks conversational.

**Game Developer**

Checks if the player character's collision mesh intersects with environmental geometry to prevent clipping through walls.

**Simulation Engineer**

Validates mechanical component interactions, ensuring two simulated parts physically overlap only when designed to connect.

**CAD Modeler**

Determines if a newly created object (like an attachment) remains entirely within the pre-defined bounds of a larger assembly.

## What Changes When You Connect

01 Accurate Overlap Data: Use the `spherical_collision` tool to instantly know if two objects overlap and exactly how deep they penetrate, eliminating guesswork in physics design.

02 Fast Ray Tracing Checks: The `ray_intersection` tool calculates precise hit points and times when a beam or projectile passes through an AABB, critical for accurate aiming mechanics.

03 Spatial Containment Validation: Use the `boundary_inclusion` tool to programmatically ensure that no part of an object accidentally exceeds its designed operational area.

04 Robust Collision Checks: The `box_collision` tool handles the common scenario of checking a sphere against a simple bounding box, speeding up general physics loop calculations dramatically.

## Real-World Applications

**Preventing Character Clipping Through Walls**

A game developer asks their agent: 'If my character (sphere) tries to move into this room's box, will they collide?' The agent runs the check using `box_collision` and reports a collision at specific coordinates, allowing the development team to adjust movement parameters instantly.

**Simulating Laser Beam Hits**

A simulation engineer asks their agent: 'What is the precise time and location where this laser beam (ray) hits the target box?' The agent uses `ray_intersection` to provide exact coordinates, which are needed for scoring or damage calculation.

**Checking Object Placement Boundaries**

A CAD modeler asks their agent: 'Is this newly attached engine (sphere) fully contained within the car chassis's bounding box?' The agent uses `boundary_inclusion` and confirms whether the object violates any defined physical limits.

**Determining Explosive Overlaps**

A physics programmer asks their agent: 'Do these two explosion spheres overlap given their centers and radii?' The agent uses `spherical_collision` to confirm the intersection, providing a penetration depth metric needed for accurate damage modeling.

# Patterns to Avoid

### Treating collision as simple boolean logic

✗ AVOID

Assuming that if two objects are close, they must be colliding. This fails when the overlap is negligible or the intersection occurs at a non-standard point.

✓ INSTEAD

Always use `spherical_collision` to get penetration depth and contact points; don't just check for proximity. Use the full metrics.

### Ignoring directional travel vectors

✗ AVOID

Trying to determine where a moving projectile hits based only on its start and end points, which is inaccurate if it passes through complex geometry.

✓ INSTEAD

Use `ray_intersection` with the specific ray origin and direction vector. This gives you the precise time of impact along the path.

### Using bounding boxes for exact shape checks

✗ AVOID

Relying solely on an AABB to confirm if a complex, curved object is inside a container. The box might be too large or miss subtle corners.

✓ INSTEAD

For containment, use `boundary_inclusion`. It accurately verifies the sphere's position against the defined bounds.

## The Right Fit

Use this MCP if your work requires verifiable, mathematical proof of spatial interaction. If you need to know *if* two objects overlap, or *where* a beam hits an object, this is your tool. Specifically, use `spherical_collision` when dealing with perfectly spherical physics bodies. Don't use it if you just need general distance checks; the MCP provides intersection metrics (depth, time of impact), which are much more detailed. If your primary need is merely to know if an object exists within a large, simple container, then `boundary_inclusion` handles that clean check better than other tools.

## Using Collision Detection Primitives MCP for 3D Physics Simulation

Right now, checking physics interactions is a massive manual task. Developers spend weeks writing complex geometric math functions to handle every scenario: sphere-to-sphere overlap, ray tracing against defined boxes, and verifying object placement within boundaries. It's all tedious copy-pasting of linear algebra that often breaks when the simulation gets too complex.

With this MCP, you just ask your agent a question about space. The engine handles the math instantly, giving you precise data like contact points or time of impact. You get usable physics results without ever touching a geometry formula.

## How Collision Detection Primitives MCP Improves Game Object Placement

The biggest manual headache is ensuring new assets don't violate the space they occupy. You currently have to manually check that a newly placed piece (like an engine block) isn't partially outside its mounting area, or you risk runtime errors and broken simulations.

Now, simply ask your agent if the object falls within bounds using `boundary_inclusion`. It gives a definitive pass/fail answer, guaranteeing structural integrity every single time.

# Collision Detection Primitives: 4 Tools for Spatial Math and Intersection Checking

These tools allow your agent to check overlaps between spheres, AABBs, rays, and planes, giving you critical physical data in a conversational format.

| # | TOOL | DESCRIPTION |
|---|---|---|
| 01 | `boundary_inclusion` | Verifies whether a sphere is fully contained inside specific geometric boundaries defined by minimum and maximum coordinates. |
| 02 | `box_collision` | Calculates if a sphere overlaps or collides with an axis-aligned bounding box, providing key collision details. |
| 03 | `ray_intersection` | Checks for intersections between a projected ray and an AABB, returning the time of impact along the path. |
| 04 | `spherical_collision` | Determines if two spheres are colliding and returns critical data like penetration depth and contact point. |

# See It in Action

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

U I have two virtual objects. One is a sphere at (0, 0, 0) with radius 5, and the other is centered at (8, 0, 0) with radius 4. Are they overlapping?

**Collision Check Result:**

- **Overlap Status:** Yes, collision detected.
- **Penetration Depth:** 1.0 unit.
- **Contact Point:** (5, 0, 0).

The objects intersect exactly at these coordinates. You can use this data to trigger an impact event.

U A projectile is fired from (-20, 10, 5) heading toward the box defined by min(-1, 0, -1) and max(1, 2, 1). When does it hit?

**Ray Intersection Analysis:**

- **Target AABB:** Defined.
- **Hit Detected:** Yes.
- **Time of Impact (t):** 4.0 units.
- **Impact Coordinates:** (-1, 1, 0).

The projectile hits the boundary at time t=4.0.

U Check if my main character's sphere is entirely within the map boundaries (min: -5, -5, -5; max: 5, 5, 5).

**Boundary Inclusion Report:**

- **Sphere Center:** (2.1, 0, 3).
- **Radius:** 1.5.
- **Is Contained? YES**. The sphere is fully within the specified boundaries.

The object placement is safe.

# Frequently Asked Questions

01 **How do I use Collision Detection Primitives MCP to check if two items are hitting in a game?**

You tell your agent the coordinates and radii of both spheres. The MCP runs the specialized collision check, giving you an instant answer on whether they overlap, along with precise data like penetration depth so you can model realistic damage.

02 **Can I use this MCP to check if a beam hits a defined box in my simulation?**

Yes. You provide the ray's start point and direction, plus the AABB boundaries. The tool calculates the exact time of impact (t) and provides the coordinates where the hit occurs, which is vital for accurate targeting systems.

03 **What if I need to make sure a newly spawned object stays inside the map's limits?**

Use the boundary inclusion tool. You specify the sphere and the boundaries (min/max coordinates). It confirms with certainty that the object is entirely contained, preventing physics errors when spawning assets.

04 **Is this MCP better than writing custom collision math code?**

Absolutely. Writing custom geometry math is prone to edge-case bugs and takes huge amounts of time. This MCP handles the complex, standardized calculations for you through simple prompts, making your agent do the hard work.

05 **Can Collision Detection Primitives MCP handle different types of shapes?**

It specializes in core primitives: spheres, AABBs, rays, and planes. It's built for high-precision checks involving these foundational 3D geometry types.

# 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 `"collision-detection-primitives": { "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

# Collision Detection Primitives 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

Vinkius is an independent platform and is not affiliated with, endorsed by, sponsored by, verified by, or otherwise authorized by Collision Detection Primitives. All third-party trademarks, logos, and brand names are the property of their respective owners. Their use in this document is strictly for informational purposes to identify service compatibility and interoperability.

## DOCUMENT INFORMATION

| | |
|---|---|
| Generated | July 2026 |
| MCP Server | Collision Detection Primitives MCP |
| Server ID | `019f1e47-3748-715c-aa3c-98d5e7ce36b0` |
| Platform | Vinkius Cloud for AI Agents |
| Endpoint | `https://edge.vinkius.com/{token}/mcp` |

## LICENSE & USAGE

This document is generated automatically by the Vinkius PDF Engine. Content reflects the MCP server configuration at the time of generation and may change as updates are deployed. For the most current information, visit vinkius.com/mcp/collision-detection-primitives.