The Democratization of Spatial Computing: A Technical and Strategic Analysis of Spline 3D

Executive Summary

The digital design landscape is currently undergoing a structural transformation, moving from static, two-dimensional interfaces toward immersive, spatial, and interactive experiences. This shift, driven by advancements in browser capabilities (WebGL, WebGPU) and hardware acceleration, has created a demand for tools that bridge the gap between high-fidelity 3D content creation (DCC) and real-time web deployment. Historically, this domain was bifurcated: artists used complex desktop software like Blender or Maya to create assets, while developers used code-heavy libraries like Three.js to render them. This workflow created a significant "delivery gap," characterized by friction, data loss, and high technical barriers to entry.

Spline 3D has emerged as a pivotal platform addressing this specific disconnect. By positioning itself not merely as a modeling tool but as an end-to-end *interactive delivery system*, Spline effectively democratizes the production of 3D web experiences. It offers a browser-based, collaborative environment that synthesizes vector design principles with 3D geometry, prioritizing interactivity and real-time rendering over the offline photorealism of traditional DCC tools.

This report provides an exhaustive analysis of the Spline platform as of late 2025. It examines the platform's core architecture, including its cloud-native engine and proprietary runtime environments for Web, iOS (Metal), and Android (Vulkan). It analyzes the recent bifurcation of its interface into the standard 3D editor and "Hana," a 2D/3D hybrid canvas designed for product interface design. Furthermore, it explores the integration of generative AI through "Spell," evaluating the implications of Gaussian Splatting and large world models on the future of asset creation.

Through a detailed technical review of its physics engines, game logic systems, and platform integrations, this analysis suggests that Spline is evolving into a foundational authoring tool for the Spatial Web. While it faces challenges regarding performance optimization on low-end devices and competition from established game engines, its "No-Code" philosophy and tight integration with frontend frameworks like React and Webflow position it uniquely to capture the growing market of designers transitioning into spatial computing.

---

1. The Paradigm Shift in Digital Design Tools

1.1 Historical Context: From Flash to WebGL

To understand Spline's significance, one must contextualize it within the history of web interactivity. In the early 2000s, Adobe Flash reigned supreme as the primary vehicle for interactive, rich media on the web. It offered a unified authoring environment where designers could draw, animate, and script interactions without deep engineering knowledge. However, Flash was a proprietary plugin, a "black box" that existed outside the open web standards, leading to its eventual demise with the rise of mobile devices and HTML5.

The post-Flash era saw the rise of WebGL (Web Graphics Library), a JavaScript API for rendering interactive 2D and 3D graphics within any compatible web browser without the use of plug-ins. While powerful, WebGL is a low-level API; drawing a simple cube requires significant boilerplate code involving shaders, buffers, and matrix math. Libraries like Three.js abstracted much of this complexity, yet they remained code-first tools. A designer could not simply "draw" in Three.js; they had to describe geometry through syntax.

1.2 The "Figma-tization" of 3D

Spline enters the market at a moment defined by the success of Figma—a tool that proved high-performance design software could run in the browser and that real-time collaboration (multiplayer) fundamentally changes how teams work. Spline adopts this "Figma-tization" philosophy and applies it to the Z-axis.

The platform eliminates the traditional "waterfall" workflow of 3D design (Model \$\rightarrow\$ Texture \$\rightarrow\$ Rig \$\rightarrow\$ Export \$\rightarrow\$ Code) and replaces it with a simultaneous, iterative process. A designer can tweak the bevel of a 3D button while a developer adjusts the React props controlling its state, all within the same URL. This architectural choice is not just a convenience; it is a structural change in the production pipeline, collapsing the distinction between "asset" and "implementation."

1.3 The Browser as an Operating System

Spline's existence is a testament to the maturity of the browser as an application runtime. By leveraging WebAssembly (Wasm) and advanced shader compilation, Spline delivers a viewport performance that rivals native desktop applications for mid-fidelity work. This accessibility allows it to bypass IT procurement friction in large organizations—there is no software to install, no license keys to manage locally, and no heavy hardware dependency for the initial viewing of files. This accessibility is a strategic wedge, allowing Spline to permeate marketing and product teams that would never approve a seat license for Cinema 4D.[1]

---

2. Core Architecture and The Runtime Engine

2.1 Cloud-Native Infrastructure

At its foundation, Spline operates entirely within the web browser. Unlike Blender, which relies on local computation, Spline's architecture is client-server based. The "truth" of the scene creates a Single Source of Truth (SSOT) stored on Spline's servers. When a user edits a mesh, operations are calculated locally for immediate feedback but synchronized via WebSockets to the server and other connected clients.

This infrastructure supports the platform's robust multiplayer capabilities. Multiple users can manipulate objects, adjust lighting, and tweak interaction logic simultaneously within the same scene. This synchronous editing capability addresses a chronic pain point in traditional 3D pipelines, where file versioning (e.g., final_v3_revised.blend) and large file transfers often stifle iteration. In Spline, presence indicators show where team members are looking, and comments can be pinned directly to 3D coordinates on an object, contextually anchoring feedback to geometry.[2]

2.2 The Runtime Engine vs. The Editor

It is crucial to distinguish between Spline the *Editor* and Spline the *Runtime*.

- **The Editor:** This is the SaaS application used for authoring. It is a React-based interface overlaying a WebGL canvas. It handles the overhead of UI, history states (undo/redo), and collaborative data streams.
- **The Runtime:** This is the lightweight engine that plays the content on the final website or app. When a user "exports" a scene, they are generating a bundle that includes the scene data (JSON/binary) and this runtime loader.

The runtime is highly optimized for delivery. It handles the render loop, the camera interpolation, physics calculations, and event listening. By controlling the runtime, Spline ensures that what the designer sees in the editor is pixel-perfect to what the end-user sees in production. This contrasts with the GLTF workflow, where a model might look different in Blender's viewport versus a Three.js scene due to differences in lighting models or material interpretation.[4]

2.3 Native Runtimes: Beyond the WebView

A critical technical achievement in 2025 is Spline's expansion beyond WebGL into native mobile rendering. While WebGL allows 3D in mobile browsers, it incurs significant overhead when embedded in native apps via WebViews (effectively running a browser inside an app).

iOS Runtime (Metal)

Spline has developed a native runtime for iOS that utilizes **Metal**, Apple's low-level, low-overhead hardware-accelerated 3D graphic and compute shader API.

- **Performance Gains:** By bypassing the browser stack, the Metal renderer offers significantly higher frame rates and lower battery consumption. It allows for more complex geometry and expensive shaders (like transmission and subsurface scattering) that would cause thermal throttling in a mobile web view.[5]
- **Integration:** The export generates a .splineswift file and provides a Swift package. Developers instantiate a SplineView component within SwiftUI, which loads the scene. This allows the 3D content to interact with native iOS UI elements seamlessly.

Android Runtime (Vulkan)

Similarly, Spline offers a runtime for Android powered by **Vulkan**, the cross-platform, high-performance API.

- **Mechanism:** This runtime can interpret Spline scenes and render them on the vast array of Android devices. It supports exporting as an Android App Bundle (AAB) or APK directly, effectively turning Spline into a no-code Android app builder for simple 3D applications.[7]
- **Compatibility:** For older devices that do not support Vulkan, the runtime likely creates a fallback to OpenGL ES, ensuring broad compatibility while prioritizing performance on modern hardware.

---

3. The Creative Interface: Standard Editor and Hana

Spline has bifurcated its interface to serve two distinct mental models: the traditional 3D modeler and the 2D product designer.

3.1 The Standard 3D Editor

The standard editor is the core of Spline, designed for spatial arrangement and modeling.

- **Parametric Modeling:** Users start with primitives (cubes, spheres, torus) that remain mathematically defined. Parameters like "corner radius," "extrusion depth," and "slice" allow for non-destructive editing. This is familiar to UI designers accustomed to vector tools, where a rectangle is defined by properties rather than vertices.[2]
- **Polygonal Modeling:** Spline includes essential mesh editing capabilities. Users can enter "Edit Mode" to manipulate vertices, edges, and faces. Features like loop cuts, face extrusion, and inset allow for the creation of custom geometry, though the toolset is simplified compared to Blender.
- **Sculpting:** A sculpting mode allows for organic shape manipulation using brushes (Grab, Inflate, Smooth). This is particularly popularized by the "Claymorphism" trend in 3D web design, where soft, blobby shapes are preferred over hard-surface realism.

3.2 Hana: The Canvas for Interactivity

Hana represents Spline's most aggressive move into the UI/UX design space. It is described as a "canvas for interactivity" and operates as a hybrid 2D/3D environment.

- **Vector Networks:** Hana incorporates 2D vector editing tools that are surprisingly robust. Unlike standard SVG paths, Spline's vector networks allow multiple lines to connect to a single node, a feature pioneered by Figma. These networks can exist on a 2D plane or be extruded into 3D volumes.[9]
- **Infinite Canvas & Auto Layout:** Hana treats the workspace as an infinite board where users can arrange "Frames." It supports Auto Layout (similar to CSS Flexbox), allowing objects to automatically resize and reposition based on content. This is critical for designing responsive 3D interfaces that must adapt to different screen sizes.[10]
- **Diegetic UI Construction:** Hana excels at creating "screens within screens." Designers can build a 2D interface using vectors and text, and then project that entire frame onto a 3D curved surface (like a smartwatch screen) using **3D Projection** features.

3.3 Materiality and The "Apple Aesthetic"

Spline's rendering engine is tuned for Non-Photorealistic Rendering (NPR), specifically the clean, tactile aesthetic prevalent in modern SaaS marketing.

- **Layer-Based Shading:** Materials are built using a stack, similar to Photoshop layers. A single object can have a Base Color layer, a Lighting layer (Phong/Lambert/Physical), a Matcap layer, and a Gradient layer. This approach is more intuitive for 2D designers than the node-based shader graphs found in Blender or Unreal Engine.
- **Liquid Glass:** A standout feature in Hana is the "Liquid Glass" effect. It simulates real-time refraction and chromatic aberration. This allows for the creation of "frosted glass" (Glassmorphism) effects where background elements are blurred and distorted through the foreground object. Achieving this in raw WebGL usually requires complex shader coding; Spline exposes it as a single toggle.[11]
- **Noise & Distortion:** Procedural noise layers allow designers to add texture and imperfection without relying on heavy image files, preserving the lightweight nature of the web export.

3.4 Lighting and Post-Processing

Spline supports standard lighting types (Directional, Point, Spot, Ambient). However, it relies on real-time approximations rather than ray tracing. Shadows are often achieved through shadow maps or "Soft Shadow" baking.

- **Post-Processing:** The camera object supports a stack of effects including Bloom, Chromatic Aberration, Vignette, and Depth of Field (DoF). These screen-space effects add a "cinematic" polish to the render, masking the low-poly nature of the geometry.[3]

---

4. Interaction Design and Logic Systems

The defining characteristic of Spline is that it produces *interactive* systems, not just static images.

4.1 The Event-Action Paradigm

Interaction logic in Spline is constructed using a cause-and-effect model, accessible via the "Events" panel.

- **Events (Triggers):** These are the inputs that initiate logic. They include:
  - **Mouse/Touch:** MouseDown, MouseUp, MouseHover, Click.
  - **Keyboard:** KeyDown, KeyUp, KeyPress (mapped to specific keys).
  - **Scene:** Start (on load), LookAt (camera orientation), Follow (object tracking).
  - **Sensors:** Collision (physics interaction), Distance (proximity), ScreenResize.[3]
- **Actions (Responses):** These are the outputs.
  - Transition: Interpolates properties (position, scale, color) from the current state to a target state.
  - Play Sound: Triggers audio files.
  - Open URL: Navigates the browser.
  - Console Log: Debugging aid.
  - Physics Impulse: Applies force to a rigid body.

4.2 Animation Systems: State vs. Timeline

Spline offers two distinct animation workflows to cater to different complexity levels.

State-Based Animation

This system mirrors CSS transitions or prototyping tools like Principle.

- **Mechanism:** A user defines a "Base State" and creates a new "State 1" where the object is modified (e.g., rotated 90 degrees). An event triggers the interpolation between Base and State 1.
- **Spring Physics:** Transitions can use spring curves (Mass, Stiffness, Damping) rather than linear easing. This imparts a natural, elastic feel to UI elements.
- **Limitations:** This system is linear and binary. It struggles with complex sequences (e.g., Object A moves, then Object B rotates, then Object A returns).[13]

Timeline Animation

Introduced to address the limits of States, the Timeline feature brings a robust keyframe workflow.

- **Dope Sheet & Graph Editor:** Users can manipulate keyframes on a timeline tracks. A Graph Editor allows for the precise adjustment of Bezier curves for easing, granting total control over the timing and feel of the motion.[14]
- **Hybrid Triggering:** Crucially, timeline animations are integrated into the Event system. A "Game Control" event or a button click can trigger a specific Timeline sequence. This

allows for "cinematic interactions"—for example, a user clicks a button (Interaction) which triggers a complex camera fly-through of a product (Timeline Animation).

4.3 Game Controls and Physics Engine

Spline includes a robust physics simulation engine, enabling the creation of mini-games and physics-based UI.

Physics Simulation

- **Rigid Bodies:** Objects can be assigned a Body Type.
  - **Dynamic:** Affected by gravity and forces.
  - **Positioned (Kinematic):** Unaffected by gravity but can push dynamic objects (e.g., moving platforms).
- **Parameters:** Users can tweak Friction (slipperiness), Bounciness (restitution), Mass, and Damping (air resistance).[15]
- **Performance Optimization:** Physics calculations are CPU-intensive. Spline allows users to define a "Collider" shape (Sphere, Box, Capsule) that is separate from the visible geometry. This is a standard game development optimization—a complex character mesh uses a simple capsule collider for physics calculations.[16]

Game Controls Event

The **Game Controls** event is a high-level abstraction that creates a character controller instantly.

- **Input Mapping:** It automatically maps WASD and Arrow keys to movement vectors relative to the camera.
- **Mobile Virtual Controls:** For touch devices, it automatically generates on-screen virtual joysticks and buttons, solving a major UX hurdle for mobile web games.
- **Camera Follow:** The event includes sophisticated camera tracking logic (Damped follow), allowing the camera to smoothly trail the character.
- **Navigation Mesh:** A sophisticated feature usually found in engines like Unity, the "Click-to-Move" option generates a nav-mesh on walkable surfaces, allowing for point-and-click movement logic.[16]

4.4 Variables and Conditional Logic

To move beyond simple triggers, Spline introduced **Variables** (Number, Boolean, String).

- **State Management:** Variables act as the memory of the scene. A "Score" variable can increment on collision events; a "Toggle" boolean can track if a menu is open.
- **Data Binding:** Variables can be bound to Text objects, allowing for dynamic scoreboards or data visualization.
- **Logic:** While not a full scripting language, the system allows for basic conditional checks (e.g., "If Score > 10, Transition to Win State").[18]

5. Artificial Intelligence and "Spell"

Spline has aggressively integrated Generative AI under the feature set named **Spell**, moving from explicit modeling to intent-based generation.

5.1 Generative Assets

- **Text-to-3D:** Users can input prompts (e.g., "a low-poly cyberpunk car") and Spell generates a 3D mesh. While often lacking perfect topology, these are invaluable for rapid prototyping and "blocking out" scenes.[19]
- **Texture Generation:** Spell can generate seamless textures based on text prompts, which can be immediately applied as material layers (e.g., "distressed denim fabric").
- **Style Transfer:** This acts as a smart filter, using AI to re-texture or re-light a scene to match a specific artistic style (e.g., "make it look like a claymation movie").[19]

5.2 Gaussian Splatting and World Models

The most technically advanced feature is the **Spell 3D World Model**.

- **Gaussian Splatting:** This is a novel rendering technique where a scene is represented not by polygons, but by a cloud of "splats"—3D ellipsoids that carry color and opacity data. When viewed, they blend to form a photorealistic image.
- **Workflow:** Users can upload a video or a series of photos of a real-world object or environment. Spell processes this data to create a navigable 3D volume.
- **Implications:** This technique excels at capturing complex real-world details (like fur, transparency, or reflections) that are incredibly difficult to model manually. By integrating Splats, Spline positions itself as a tool for "Reality Capture" editing, allowing designers to combine scanned real-world environments with synthetic UI elements.[20]

6. Platform Integration and Ecosystem

Spline's utility is defined by its ability to integrate into external development stacks.

6.1 React Integration (@splinetool/react-spline)

For the React ecosystem, Spline provides a dedicated component wrapper that facilitates deep communication between the React DOM and the WebGL canvas.

Component Architecture:

JavaScript

```
import Spline from '@splinetool/react-spline';

export default function App() {
 function onLoad(spline) {
  // Access the Spline API instance
  spline.findObjectByName('Cube');
 }

 return (
  <Spline
   scene="https://prod.spline.design/scene.splinecode"
   onLoad={onLoad}
   onSplineMouseDown={(e) => console.log('Clicked object:', e.target.name)}
  />
 );
}
```

- **Two-Way Binding:** The integration allows for bidirectional control. React can trigger Spline events (e.g., clicking an HTML button rotates the 3D car), and Spline can trigger React functions (e.g., clicking the 3D car opens a React modal).
- **Lazy Loading:** To protect Core Web Vitals (specifically LCP - Largest Contentful Paint), the library supports lazy loading. Developers can display a lightweight placeholder image and only initialize the heavy WebGL context when the component enters the viewport.[22]

6.2 Webflow Integration

Spline has a strategic partnership with Webflow, the leading no-code web builder.

- **Spline Scene Element:** Webflow offers a native "Spline Scene" element in its builder.
- **Interaction Mapping:** Designers can tie Webflow's native IX2 interaction engine to Spline. For example, a "Page Scroll" trigger in Webflow can drive the timeline of a Spline scene. This allows for precise "Scrollytelling" experiences where a 3D product explodes or rotates in perfect sync with the user's scroll position, without writing custom JavaScript listeners.[23]

6.3 Code Export Analysis

Developers often ask about the quality of Spline's code export.

- **The "Black Box" Reality:** The default export is a URL or a binary file (.splinecode) loaded by the runtime. This is a "black box" asset. The developer cannot easily "tree-shake" unused parts of the scene or modify the geometry source code.
- **Generated Code:** Spline can export "Vanilla JS" or "React" code snippets. However, this

code heavily relies on the Spline runtime library to interpret the scene data. It does *not* export a clean, imperative Three.js scene graph that a developer can refactor manually.

- **Trade-off:** The choice is between **Ease of Integration** (Spline Runtime) and **Total Control** (Exporting as GLTF and rebuilding in React Three Fiber). The latter offers more control over the render loop and performance but discards all the interaction logic built in Spline.[25]

---

7. Performance, Optimization, and Constraints

Web-based 3D is constrained by the hardware limitations of the end-user's device, which may range from a high-end gaming PC to a budget Android phone on a 4G network.

7.1 The Performance Budget

Spline imposes soft limits to ensure performant web experiences.

- **Polygon Count:** Spline recommends keeping scenes under 100k-150k polygons for general web use. High-poly CAD models imported from SolidWorks or Rhino must be decimated (simplified) before import to prevent browser crashes.[26]
- **Texture Size:** Large textures (4k/8k) consume massive amounts of GPU memory and bandwidth. Spline automatically compresses textures (likely using formats like KTX2 or Basis Universal), but users are advised to use 2k textures or smaller.
- **Draw Calls:** Every separate object with a unique material creates a "draw call" to the GPU. Too many draw calls act as a bottleneck for the CPU. Spline allows merging geometries to reduce this count, but objects that need to move independently must remain separate.

7.2 The Performance Panel

To aid optimization, Spline includes a **Performance Panel** that runs an audit on the scene before export.

- **Metrics:** It provides real-time data on Polygon Count, Draw Calls, Texture Memory, and Estimated Load Time.
- **Geometry Compression:** The export settings include a "Geometry Quality" toggle. Setting this to "Performance" applies aggressive mesh compression (likely Draco compression), significantly reducing file size at the cost of a slight CPU penalty during decompression.[27]

7.3 Common Issues and Bugs

Community feedback highlights several recurring stability issues inherent to browser-based 3D.

- **Z-Fighting & Transparency:** Spline’s rasterizer sometimes struggles with depth sorting when multiple transparent objects overlap (e.g., glass behind glass), leading to visual artifacts or "popping."

- **Static/Glitches:** Users have reported "static" or noise on surfaces, often related to shadow map resolution or specific material combinations (e.g., Depth material) interacting poorly with certain browser WebGL implementations.[29]
- **Browser Crashes:** Extremely heavy scenes can cause the browser tab to crash due to "Out of Memory" errors, particularly on iOS Safari which has strict memory limits for web tabs.

---

8. Market Position, Pricing, and Community

8.1 Pricing Strategy and Feature Gating

Spline's pricing model reflects a strategy to capture both the educational market and the enterprise sector.

Tier	Price	Key Features	Analysis
Free	$0/mo	Unlimited public files, Watermarked exports	Excellent for students/hobbyists. The "public file" model builds the community library.
Starter	$12/mo	No watermark, Video/Audio uploads	Targeted at freelancers and portfolio sites.
Professional	$20/mo	**Variables**, **Apple/Android Exports**, **Code Export**, Unlimited Scenes	The standard for commercial production. Gating *Variables* here signals that complex logic is a premium feature.
Team	Custom	SSO, Centralized Billing	Enterprise compliance features.

Controversy: There has been community friction regarding "silent" changes to file limits on the Free plan, with users reporting an inability to create new files after hitting a low cap (e.g., 2 personal files). This tension highlights the delicate balance Spline must maintain between freemium growth and monetization.[30]

8.2 Community and Learning Ecosystem

The steep learning curve of 3D is mitigated by a vibrant community ecosystem.

- **Remix Culture:** The **Community Platform** allows users to publish scenes as "Public," enabling others to open, inspect, and fork ("Remix") them. This acts as an open-source library for 3D mechanics.[31]
- **Education:** Official documentation is highly regarded for its clarity. Furthermore, third-party influencers like **The Motion Visual (Connor)** have built extensive tutorial libraries (hundreds of videos), acting as an external onboarding engine for the platform.[32]

8.3 Competitor Analysis

- **Vs. Blender:** Blender is a production powerhouse for creating assets (modeling, UVs, complex rendering). Spline cannot compete on fidelity or modeling tool depth. However, Blender cannot natively deliver interactive web content. The workflow is often **Blender (Create) $\rightarrow$ Spline (Interact & Publish)**.[33]
- **Vs. Three.js / React Three Fiber (R3F):** R3F offers developers absolute control and is free (open source). However, it requires significant coding skill. Spline charges for the convenience of a GUI. For simple scenes, Spline is faster; for complex, data-driven applications, R3F remains the developer standard.[34]
- **Vs. Rive:** Rive is the 2D equivalent (vector animation runtime). As Rive adds 3D features and Spline adds 2D features (Hana), they are converging on the same market: "Interactive Design Runtimes."

---

9. Conclusion

Spline 3D has successfully carved out a new category in the design tool stack. It is not trying to replace Blender for asset creation, nor is it trying to replace Unity for AAA game development. Instead, it addresses the "Interactive Web" niche—marketing sites, product configurators, and micro-games—where speed of iteration and ease of deployment are paramount.

The introduction of native mobile runtimes (Metal/Vulkan) and the hybrid 2D/3D capabilities of Hana suggest a strategic pivot toward **Spatial Computing**. As interfaces move from flat screens to spatial overlays (via Vision Pro, Quest, or AR mobile apps), Spline positions itself as the "Figma for Spatial Design."

For design teams, the value proposition is clear: Spline allows product designers to create shipping-quality 3D interactions without needing a graphics engineer. For developers, it offers a managed runtime that handles the heavy lifting of WebGL, albeit at the cost of control. While performance constraints and browser limitations remain valid concerns, Spline's trajectory suggests it will become a standard utility in the modern frontend stack, democratizing the third dimension for the next generation of the web.

Works cited

1. The Best Web Design Tools of 2025: The Top 10 - Lightflows, accessed December 20, 2025, https://www.lightflows.co.uk/blog/the-best-web-design-tools-of-2025-the-top-10/
2. 3D Design tool in the browser with real-time collaboration - Spline, accessed December 20, 2025, https://spline.design/3d-design
3. Spline Documentation: Getting started, accessed December 20, 2025, https://docs.spline.design/
4. Spline AI (Alpha): Easy 3D Design and Collaboration Software - Deepgram, accessed December 20, 2025, https://deepgram.com/ai-apps/spline-ai-alpha
5. Design and export native 3D for iOS - Spline, accessed December 20, 2025, https://spline.design/ios
6. Native 3D Embeds for iOS - Spline Documentation, accessed December 20, 2025, https://docs.spline.design/exporting-your-scene/apple-platform/native-3d-embeds-for-i-os
7. Design and export native 3D for Android - Spline, accessed December 20, 2025, https://spline.design/android
8. Native 3D Embeds for Android - Spline Documentation, accessed December 20, 2025, https://docs.spline.design/exporting-your-scene/android/native-3d-embeds-for-android
9. What is Hana? - Spline Documentation, accessed December 20, 2025, https://docs.spline.design/hana-a-canvas-for-interactivity/what-is-hana
10. Designing in Hana - Spline Documentation, accessed December 20, 2025, https://docs.spline.design/hana-a-canvas-for-interactivity/designing-in-hana
11. Effects in Hana - Spline Documentation, accessed December 20, 2025, https://docs.spline.design/hana-a-canvas-for-interactivity/effects-in-hana
12. Screen Resize Event - Spline Documentation, accessed December 20, 2025, https://docs.spline.design/interaction-states-events-and-actions/events/screen-resize-event
13. How state-based animation works - Spline Documentation, accessed December 20, 2025, https://docs.spline.design/interaction-states-events-and-actions/how-state-based-animation-works

14. Introducing 3D Timeline Animation - Spline Blog, accessed December 20, 2025, https://blog.spline.design/introducing-3d-timeline-animation
15. Physics - Spline Documentation, accessed December 20, 2025, https://docs.spline.design/designing-in-3-d/physics
16. Game Controls Event - Spline Documentation, accessed December 20, 2025, https://docs.spline.design/interaction-states-events-and-actions/events/game-controls-event
17. Physics - Create a 3D site with game controls in Spline - Design+Code, accessed December 20, 2025, https://designcode.io/spline2-physics/
18. Variables - Spline Documentation, accessed December 20, 2025, https://docs.spline.design/interaction-states-events-and-actions/variables
19. Spline AI 3D Generation – The power of AI for the 3rd dimension., accessed December 20, 2025, https://spline.design/ai-generate
20. Spell AI 3D Worlds - Spline Documentation, accessed December 20, 2025, https://docs.spline.design/spline-ai/spell-ai-3d-worlds
21. Spell by Spline - 3D World model, accessed December 20, 2025, https://blog.spline.design/introducing-spell
22. splinetool/react-spline: React component for Spline scenes. - GitHub, accessed December 20, 2025, https://github.com/splinetool/react-spline
23. Add a Spline scene to your Webflow site, accessed December 20, 2025, https://help.webflow.com/hc/en-us/articles/33961236080787-Add-a-Spline-scene-to-your-Webflow-site
24. Easily Integrate 3D on Webflow with Spline Tool → - Digidop, accessed December 20, 2025, https://www.digidop.com/blog/how-to-integrate-3d-spline-webflow
25. Exporting as Code - Spline Documentation, accessed December 20, 2025, https://docs.spline.design/exporting-your-scene/web/exporting-as-code
26. CAD to Spline: Optimization Best Practices, accessed December 20, 2025, https://docs.spline.design/importing-content/cad-to-spline-optimization-best-practices
27. How to optimize your scene - Spline Documentation, accessed December 20, 2025, https://docs.spline.design/exporting-your-scene/how-to-optimize-your-scene
28. Some Spline Tips Follow for more #3d #spline3d #spline #3ddesign #designtips - YouTube, accessed December 20, 2025, https://www.youtube.com/shorts/Aq2m0D1QTm0
29. Static issue : r/Spline3D - Reddit, accessed December 20, 2025, https://www.reddit.com/r/Spline3D/comments/1c2cbqd/static_issue/
30. Did Spline suddenly lock creating new files? : r/Spline3D - Reddit, accessed December 20, 2025, https://www.reddit.com/r/Spline3D/comments/1oqchud/did_spline_suddenly_lock_creating_new_files/
31. Community Platform - Spline Documentation, accessed December 20, 2025, https://docs.spline.design/basics/community-platform
32. LEARN 3D Design + Animation in 2025! - YouTube, accessed December 20, 2025, https://m.youtube.com/shorts/JjF3xA6eGzI

33. Question: Difference between Spline 3D vs. Blender?, accessed December 20, 2025, https://www.reddit.com/r/blender/comments/12d7hst/question_difference_between_spline_3d_vs_blender/
34. Three.js, Three-Fiber, and Spline — Which should you use?, accessed December 20, 2025, https://javascript.plainenglish.io/three-js-three-fiber-spline-which-is-best-6989f596d3a4