Vue.js and Three.js: A Blueprint for Advanced Web Graphics

The landscape of modern web development is increasingly characterized by the demand for sophisticated and engaging user experiences. Two technologies stand out as powerful enablers in this realm: Vue.js, a progressive JavaScript framework renowned for its intuitive component-based architecture and ease of integration, and Three.js, a versatile cross-browser JavaScript library that leverages WebGL to render stunning 3D graphics directly within the browser.[1] As user expectations for visually rich and interactive web content continue to rise [2], the synergistic combination of Vue.js and Three.js offers a robust foundation for building advanced web applications with immersive 3D graphics.[21] This report delves into the advanced techniques for integrating these two powerful libraries, exploring architectural patterns, state management within 3D environments, sophisticated rendering and animation methods, strategies for handling user interaction, the utilization of advanced Three.js features, and crucial considerations for optimizing performance.

Foundational Integration Patterns

The integration of Vue.js and Three.js in web development projects can be approached through several fundamental patterns, each offering distinct advantages and trade-offs.

Directly Embedding Three.js within Vue Components

One of the most fundamental ways to integrate Three.js within a Vue application involves directly instantiating and managing Three.js elements within the lifecycle hooks of Vue components.[24] This typically entails creating instances of the Three.js scene, camera, and renderer within the mounted hook of a Vue component, where the component's DOM is available.[24] To render the Three.js scene within the Vue component's template, a reference to the HTML canvas element is obtained using Vue's ref attribute.[24] This canvas reference is then passed to the Three.js WebGLRenderer to establish the rendering context.[24] The unmounted hook is then utilized to dispose of Three.js resources, preventing memory leaks.[24] This direct method grants developers granular control over every aspect of the Three.js scene and its rendering process, allowing for highly customized implementations and seamless integration with existing Three.js codebases. However, it necessitates manual management of the Three.js object lifecycle and careful consideration of how Three.js interacts within Vue's declarative paradigm.

Leveraging Vue's Composition API for Three.js Logic

Vue's Composition API offers a powerful mechanism for organizing and reusing logic through composable functions.[29] This paradigm is highly beneficial for integrating Three.js, as it allows for the creation of custom Vue composables that encapsulate the setup and management of core Three.js elements like the scene, camera, and renderer.[32] These composables can handle the initialization of Three.js objects within Vue's onMounted hook and ensure proper disposal within the onUnmounted hook, effectively managing the lifecycle of Three.js resources within the Vue component's scope.[30] By extracting Three.js related logic into composables, Vue components become more focused on their rendering and data-binding responsibilities, leading to improved code organization, testability, and maintainability, especially in larger projects.[29] This approach promotes a cleaner and more modular codebase for integrating Three.js functionality within Vue applications.

Exploring Component Libraries: TresJS and Drie

To further simplify the integration of Three.js into Vue applications, dedicated component libraries like TresJS and Drie have emerged. TresJS is a custom renderer specifically designed for Vue 3, enabling developers to build 3D scenes declaratively using Vue components directly within their templates.[21] It provides a comprehensive set of Vue components that mirror fundamental Three.js functionalities, such as <TresCanvas> for the main rendering area, <TresPerspectiveCamera> for defining the viewpoint, and <TresMesh> along with geometry components like <TresBoxGeometry> for creating 3D objects.[41] TresJS also handles canvas resizing automatically and integrates a render loop, making it highly convenient for Vue developers to work with Three.js.[27] Drie is another open-source Vue 3 component library inspired by React Three Fiber, offering a type-safe and composable approach to building 3D web applications with Three.js, leveraging TypeScript and Vue 3's Composition API.[31] These component libraries provide a high level of abstraction, significantly simplifying Three.js integration and accelerating development by offering pre-built and reactive Vue components for common 3D elements.

Approach	**Level of Abstraction**	**Ease of Use**	**Control over Three.js**	**Reusability**	**Community Support/Maturity**
Direct Embedding	Low	Moderate	High	Moderate	High
Composition API	Medium	Moderate	High	High	High

TresJS	High	High	Moderate	High	Medium
Drie	High	Moderate	Moderate	High	Low

Crafting Complex 3D Scenes and Animations with Vue

Building intricate 3D scenes and bringing them to life with compelling animations within a Vue application involves leveraging the strengths of both Vue.js and Three.js through well-defined patterns.

Building Reusable and Reactive 3D Components

A cornerstone of building complex 3D applications with Vue.js and Three.js is the creation of reusable and reactive 3D components.[21] By encapsulating specific 3D objects or functionalities within custom Vue components, developers can create modular and maintainable codebases. These components can be designed to accept parameters through Vue props, such as color, size, initial position, and animation speeds, making them highly configurable and adaptable for various use cases.[42] Vue's reactivity system plays a crucial role here, ensuring that any changes in the prop values are automatically and efficiently reflected in the corresponding Three.js objects within the 3D scene.[43] For instance, a custom <AnimatedCube> component might accept a color prop, which, when updated in the parent component's state, would automatically change the material color of the cube in the 3D scene. This component-based approach significantly enhances the organization and scalability of complex 3D web applications.

Advanced Animation Techniques: Keyframes, Tweening, and Custom Loops

Animating Three.js objects within a Vue context can be achieved through various advanced techniques. TresJS provides the useRenderLoop composable, which allows developers to create custom animation loops that synchronize with the browser's refresh rate, ensuring smooth and performant animations.[34] Within these loops, properties of Three.js objects, such as their rotation, position, and scale, can be accessed and manipulated through Vue refs obtained from TresJS components or directly managed Three.js instances.[27] Keyframe animations in Three.js offer a way to define specific values for object properties at different points in time, allowing for the creation of complex, pre-defined animation sequences that can be integrated with Vue.[7] External JavaScript animation libraries like Tween.js provide more sophisticated animation capabilities, including interpolation and easing functions, which can be used within Vue applications to control the animation of Three.js objects.[3]

Furthermore, Vue-specific animation libraries like Vue Motion offer a declarative syntax for creating a wide range of animations that can interact with or be applied to Three.js scenes.[48] The combination of Vue's reactivity and component model with Three.js's animation features and external libraries empowers developers to create intricate and visually captivating 3D animations in their web applications.

Efficient Model Loading and Management in Vue

Efficiently loading and managing 3D model assets is paramount for the performance of complex 3D web applications. Various strategies exist for loading different 3D model formats (e.g., glTF, OBJ, FBX) within a Vue application using Three.js loaders.[16] TresJS simplifies this process with its useLoader and useGLTF composables, as well as the <GLTFModel> component, which streamline the loading of various model formats, including handling asynchronous operations gracefully using Vue's <Suspense> component to display loading indicators.[34] Best practices for model optimization are crucial and include prioritizing the glTF format for its web-friendliness and efficient runtime delivery [17], utilizing Draco compression to significantly reduce file sizes (often supported directly by TresJS's useGLTF [17]), optimizing texture sizes and formats [19], and potentially using tools like gltfjsx to further optimize and prepare models for web use.[19] Once loaded, these 3D model assets can be effectively managed within Vue components, allowing developers to access their properties (e.g., meshes, materials, animations) and manipulate them according to the application's logic.[16]

State Management for Immersive 3D Environments

In the development of intricate 3D web applications using Vue.js and Three.js, especially those involving numerous interactive elements and shared data, robust and centralized state management becomes essential.

Integrating Vuex and Pinia for Centralized 3D State

For complex 3D scenes with multiple interactive elements and shared data, employing a centralized state management solution like Vuex [1] or the more modern and recommended Pinia [28] is highly beneficial. These libraries provide a centralized store for managing the application's state, ensuring data consistency and predictable updates across different components.[55] Within this centralized store, the state of the 3D scene, including camera position, object properties (like position, rotation, and material attributes), and animation parameters, can be effectively managed.[59] Different Vue components within the application can then access and update this shared 3D scene state through the mechanisms provided by Vuex (actions, mutations, getters) or Pinia (actions, state, getters).[73] This centralized approach simplifies the coordination of various parts of the 3D application, making it easier to track changes, implement complex interactions, and maintain a consistent visual representation of the 3D environment. For new projects, Pinia is generally recommended due to its simpler API and better TypeScript support.[56]

Managing Scene Objects, Camera, and Rendering Parameters

Vuex or Pinia can be effectively utilized to manage the dynamic properties of individual Three.js scene objects, such as their position, rotation, and material properties like color or texture.[73] Crucial camera parameters, including its position, the point it's looking at (target), and its field of view, can also be controlled through the reactive state managed by these state management libraries.[73] Furthermore, global rendering parameters like the scene's background color, fog settings, and shadow properties can be managed using the centralized state management system. This provides a highly reactive and efficient way to control and dynamically update various visual and structural aspects of the Three.js scene based on the application's logic, user interactions, or external data sources, without requiring direct and potentially cumbersome manipulation of Three.js objects within individual Vue components. Instead, changes to the application's state automatically trigger corresponding updates in the 3D scene, leading to a more streamlined and maintainable codebase.

Ensuring Reactive Updates in the 3D Context

Vue's powerful reactivity system, when seamlessly integrated with Three.js through libraries like TresJS or through careful manual implementation, ensures that any changes in the application's state are automatically and efficiently propagated to the rendered 3D scene, resulting in immediate visual updates.[21] This data-driven approach significantly reduces the need for manual and imperative updates to the Three.js scene whenever the application's state changes. However, it's important to be aware of potential performance implications that might arise when using Vue's reactivity system with large amounts of 3D data or frequent updates. To mitigate these concerns, strategies such as using plain JavaScript objects for frequently updated properties within the core render loop can be employed to bypass Vue's proxy overhead.[27] Leveraging Vue's reactivity simplifies the development of highly interactive and dynamic 3D experiences where changes in the underlying data are instantly reflected in the visual output, creating a more engaging and responsive user interface.

Elevating Visual Fidelity and Performance through Advanced Rendering

Achieving high-quality visuals and maintaining optimal performance in Vue.js and Three.js applications often requires the implementation of advanced rendering techniques.

Implementing Post-Processing Effects with TresJS and Native Three.js

Post-processing is a vital technique for enhancing the visual quality of rendered 3D scenes by applying various image-based effects after the initial rendering pass.[13] The @tresjs/post-processing package simplifies this process for TresJS users by providing a set of Vue components that wrap both the postprocessing library by pmndrs and native Three.js post-processing effects, allowing for the easy addition of effects like bloom, depth of field, motion blur, and glitch using a declarative component-based approach.[74] For more fine-grained control, post-processing effects can also be implemented directly using Three.js's EffectComposer class and its associated RenderPass and various effect passes within a Vue application.[13] While post-processing can significantly improve visual appeal, it's crucial to consider its performance impact and carefully choose and configure effects to achieve the desired visual enhancements without sacrificing frame rates.

Creating Custom Shaders for Unique Visuals and Optimizations

Custom shaders in Three.js, implemented using ShaderMaterial, empower developers to write their own rendering logic using GLSL (OpenGL Shading Language) to achieve unique visual effects or optimize rendering performance for specific scenarios.[8] GLSL shaders consist of a vertex shader, which manipulates the position of vertices, and a fragment shader, which determines the color of each pixel.[11] Data can be passed to custom shaders from a Vue application using uniforms (variables with the same value for all vertices/fragments) and attributes (vertex-specific data).[11] Custom shaders offer immense flexibility for creating visual effects not possible with standard Three.js materials and for optimizing rendering performance by implementing specialized rendering algorithms on the GPU.[19] While requiring knowledge of GLSL, custom shaders allow developers to push the boundaries of web graphics.

Strategies for Optimizing Rendering Performance: LOD, Instancing, and Culling

Optimizing rendering performance is critical for delivering smooth and responsive 3D web experiences. Several key strategies can be employed in Vue + Three.js applications.[19] Level of Detail (LOD) involves rendering different versions of a 3D model with varying complexity based on its distance from the camera, reducing the polygon count for distant objects.[19] Instancing is a technique for efficiently rendering a large number of identical 3D objects with a single draw call.[19] Culling techniques, such as frustum culling (discarding objects outside the camera's view) and backface culling (avoiding rendering unseen back faces), help reduce the number of objects the GPU needs to process.[19] Other optimization strategies include texture optimization, reducing draw calls by merging geometries, and potentially using web workers for offloading tasks.[19] Vue-specific optimizations like v-once and v-memo can also indirectly benefit Three.js rendering by preventing unnecessary component re-renders.[85]

Enabling Rich User Interactions in 3D

Creating engaging and immersive 3D web applications necessitates providing users with rich and intuitive ways to interact with the virtual environment.

Handling Mouse, Touch, and Keyboard Events on 3D Objects

Capturing and responding to user input events directly on the Three.js objects within the 3D scene is fundamental for creating interactive experiences.[3] This includes handling mouse clicks, mouse movements for hover effects and dragging, touch events on mobile devices, and keyboard presses. Raycasting is a crucial technique for detecting which 3D object the user is interacting with by casting an invisible ray from the input event's position into the scene.[3] Additionally, implementing camera controls like OrbitControls allows users to dynamically rotate, zoom, and pan their view of the 3D scene using mouse or touch gestures.[3] Handling keyboard input can provide further control mechanisms for the 3D scene, such as triggering specific actions or navigating the environment.[53]

Implementing Responsive Design for Varying Screen Sizes

Ensuring that Vue + Three.js applications adapt seamlessly across a wide range of screen sizes and devices is crucial for reaching a broader audience.[25] Handling the browser's resize event is essential for dynamically updating the camera's aspect ratio and the renderer's output size whenever the window dimensions change, preventing distortion and ensuring correct rendering on different displays.[25] Utilizing CSS media queries can also help adjust the layout and sizing of the HTML container holding the 3D canvas, allowing for flexible integration of the 3D scene within the overall web page layout on various screen sizes.

Developing Intuitive 3D Controls and Interfaces

Designing user interfaces specifically tailored for 3D applications is vital for creating accessible and enjoyable experiences.[3] This involves providing clear visual cues, understandable feedback, and intuitive control mechanisms that guide users effectively through the 3D environment. Utilizing GUI libraries like dat.GUI allows developers to easily create interactive panels for users to directly manipulate various parameters of the 3D scene in real-time.[3] Custom user interface elements can also be created within Vue components to control different aspects of the 3D scene through the application's state management system, providing a more integrated and branded user experience.

Extending Capabilities with Advanced Three.js Features

Three.js offers a rich set of advanced features that can significantly extend the capabilities of Vue.js-based 3D web applications.

Integrating Physics Simulations for Realistic Interactions

Integrating physics engines like Ammo.js, Cannon.js, and Rapier into Three.js scenes enables the creation of realistic physical interactions between 3D objects.[1] These engines simulate real-world physics, allowing for natural-looking object movements, collisions, and other physically based behaviors. Integration with Vue applications can be achieved through dedicated libraries or by directly managing the physics simulation loop and data synchronization within Vue components or composables.[97] Understanding concepts like rigid bodies, colliders, and forces is crucial for effectively implementing physics in a Three.js scene.

Creating Immersive VR and AR Experiences within Vue.js

The WebXR Device API provides a standard way to access virtual reality (VR) and augmented reality (AR) capabilities on compatible web browsers and devices.[24] Integrating WebXR with Vue and Three.js allows for the creation of truly immersive experiences directly within the browser.[21] This can involve utilizing community-developed libraries that provide higher-level abstractions or by directly interacting with the WebXR API for more granular control. Key WebXR concepts include initiating immersive sessions for VR and AR rendering, establishing appropriate reference spaces to define the coordinate system for the XR experience, and handling input from VR controllers or AR tracking systems. While some Vue VR libraries exist [103], direct WebXR integration offers the most flexibility and access to the latest features.

Learning from the Community: Case Studies and Open-Source Projects

Examining existing projects and learning from the experiences of other developers within the community is an invaluable resource for advancing Vue.js and Three.js skills.

Analysis of Notable Vue.js and Three.js Implementations

Numerous open-source projects and case studies showcase advanced implementations of Vue.js and Three.js.[30] Analyzing their architecture, integration patterns, and the techniques they employ for rendering, animation, and user interaction can provide valuable insights into real-world applications of these technologies. For example, projects like "Vue-Threejs-1" on GitHub demonstrate interactive 3D models with user controls built using Vue.js, VueX, and Three.js.[46] Studying such implementations can reveal effective strategies and best practices for combining these libraries.

Identifying Common Architectural Patterns and Solutions

Analyzing advanced Vue.js and Three.js projects reveals recurring architectural patterns and solutions for common challenges. Component-based scene management, where 3D elements are treated as reusable Vue components, is a prevalent pattern.[21] State management, often using Vuex or Pinia, is frequently employed to synchronize the state of 3D elements and manage interactions.[28] Common solutions for performance optimization include techniques like model optimization, instancing, and careful management of the render loop.[19] Recognizing these established patterns and solutions can streamline the development process and lead to more robust and maintainable applications.

Addressing Performance Challenges and Optimization Strategies

Achieving optimal performance is a critical aspect of developing advanced web applications that integrate Vue.js and Three.js.

Identifying Common Performance Bottlenecks in Vue + Three.js Applications

Several common performance bottlenecks can arise when combining Vue.js and Three.js.[19] Excessive re-renders triggered by Vue's reactivity system, especially when dealing with large or frequently changing 3D data, can impact performance.[27] High polygon counts in 3D models, inefficient use of materials and textures, an excessive number of draw calls to the GPU, unoptimized or overly complex animations, and performance-intensive post-processing effects can all contribute to performance issues. Understanding these potential bottlenecks early in the development process is crucial for building performant applications.

Best Practices for Efficient Resource Management and Memory Usage

Efficiently managing resources and minimizing memory usage are essential for the stability and performance of Vue + Three.js applications.[20] This includes properly disposing of Three.js objects (geometries, materials, textures) when they are no longer needed to prevent memory leaks.[20] Optimizing texture usage by using compressed formats, reducing resolution where possible, and employing texture atlases can significantly improve performance.[19] Efficiently managing and unloading 3D asset data when it's no longer required is also crucial, especially in applications that load and unload models dynamically.

Profiling and Debugging Techniques for Performance Optimization

Profiling and debugging are crucial for identifying and resolving performance issues in Vue + Three.js applications.[19] Browser developer tools offer performance analysis capabilities, while libraries like Three.js's stats.js [19] and the Vue Devtools browser extension [85] provide valuable insights into rendering performance and component behavior. Specialized WebGL debugging tools like Spector.js [19] allow for in-depth analysis of GPU operations. By utilizing these tools, developers can pinpoint performance bottlenecks, analyze rendering performance, and implement targeted optimizations to improve the application's responsiveness and frame rates.

Conclusion

The integration of Vue.js and Three.js presents a powerful paradigm for creating advanced and visually compelling web applications. By understanding and applying the techniques discussed in this report, developers can effectively leverage the strengths of both libraries to build sophisticated 3D experiences. Whether through direct embedding, the Composition API, or dedicated component libraries like TresJS and Drie, the key lies in adopting a component-based architecture, implementing robust state management, employing advanced rendering and animation techniques, providing intuitive user interactions, and diligently optimizing for performance. As web technologies continue to evolve, the synergistic power of Vue.js and Three.js will undoubtedly continue to drive innovation in the realm of interactive 3D web development.

Works cited

1. New project to explore sounds, rhythms and music created with a 3D physics engine. Using Three.js, Ammo.js, Vue 3, VueX and PrimeVUE components. Written in Typescript. - Reddit, accessed April 5, 2025, https://www.reddit.com/r/threejs/comments/lsf38j/new_project_to_explore_sounds_rhythms_and_music/
2. Create Immersive 3D Experiences with Three.js Comprehensive Guide - MoldStud, accessed April 5, 2025, https://moldstud.com/articles/p-create-immersive-3d-experiences-with-threejs-

guide
3. How to Create 3D Animations with Three.js - PixelFreeStudio Blog, accessed April 5, 2025, https://blog.pixelfreestudio.com/how-to-create-3d-animations-with-three-js/
4. Examples - Three.js, accessed April 5, 2025, https://threejs.org/examples/
5. Rendering 3D file with Three js - javascript - Stack Overflow, accessed April 5, 2025, https://stackoverflow.com/questions/64427359/rendering-3d-file-with-three-js
6. Tresjs/tres: Declarative ThreeJS using Vue Components - GitHub, accessed April 5, 2025, https://github.com/Tresjs/tres
7. The three.js Animation System, accessed April 5, 2025, https://discoverthreejs.com/book/first-steps/animation-system/
8. Animating Letters with Shaders: Interactive Text Effect with Three.js & GLSL | Codrops, accessed April 5, 2025, https://tympanus.net/codrops/2025/03/24/animating-letters-with-shaders-interactive-text-effect-with-three-js-glsl/
9. Intermediate Three.js Tutorial: Make a Globe with Custom Shaders - YouTube, accessed April 5, 2025, https://www.youtube.com/watch?v=vM8M4QloVL0
10. Three.js Shaders (GLSL) Crash Course For Absolute Beginners - YouTube, accessed April 5, 2025, https://www.youtube.com/watch?v=oKbCaj1J6EI
11. Creating a custom shader in Three.js - DEV Community, accessed April 5, 2025, https://dev.to/maniflames/creating-a-custom-shader-in-threejs-3bhi
12. Three.js Custom Shader - javascript - Stack Overflow, accessed April 5, 2025, https://stackoverflow.com/questions/13013658/three-js-custom-shader
13. Post-processing – three.js docs, accessed April 5, 2025, https://threejs.org/docs/manual/en/introduction/How-to-use-post-processing.html
14. Excluding objects / layers from post-processing in Three.js - Stack Overflow, accessed April 5, 2025, https://stackoverflow.com/questions/14837835/excluding-objects-layers-from-post-processing-in-three-js
15. Post-processing - Three.js Journey, accessed April 5, 2025, https://threejs-journey.com/lessons/post-processing
16. Three.js Tutorial: How to Load a Model - YouTube, accessed April 5, 2025, https://www.youtube.com/watch?v=yIyLefnMc1c
17. Loading 3D models – three.js docs, accessed April 5, 2025, https://threejs.org/docs/manual/en/introduction/Loading-3D-models.html
18. Loading GLTF models in Nuxt.js / Vue.js - Questions, accessed April 5, 2025, https://discourse.threejs.org/t/loading-gltf-models-in-nuxt-js-vue-js/8326
19. Building Efficient Three.js Scenes: Optimize Performance While Maintaining Quality, accessed April 5, 2025, https://tympanus.net/codrops/2025/02/11/building-efficient-three-js-scenes-optimize-performance-while-maintaining-quality/
20. The Big List of three.js Tips and Tricks!, accessed April 5, 2025, https://discoverthreejs.com/tips-and-tricks/

21. Integration with Vue 3 - threejs - Reddit, accessed April 5, 2025, https://www.reddit.com/r/threejs/comments/1i6vdzr/integration_with_vue_3/
22. TresJS - Build 3D Scenes declarative using Vue components : r/vuejs - Reddit, accessed April 5, 2025, https://www.reddit.com/r/vuejs/comments/1c6glfy/tresjs_build_3d_scenes_declarative_using_vue/
23. TresJS, accessed April 5, 2025, https://tresjs.org/
24. Rotating cube made using Three.js and Vue 3 - YouTube, accessed April 5, 2025, https://www.youtube.com/watch?v=tYqrwrnDonA
25. Building an interactive web portfolio with Vue + Three.js — Part Three: Implementing Three.js | by Máximo Fernández | NicaSource | Medium, accessed April 5, 2025, https://medium.com/nicasource/building-an-interactive-web-portfolio-with-vue-three-js-part-three-implementing-three-js-452cb375ef80
26. How to implement Three.js scene example in Vue 3? · vuejs · Discussion #9057 - GitHub, accessed April 5, 2025, https://github.com/vuejs/core/discussions/9057
27. How to add 3D to your Vue App using ThreeJS - YouTube, accessed April 5, 2025, https://www.youtube.com/watch?v=1mFWG8WBif8
28. Rendering three.js canvas in VUE without using let variables outside nor appendChild, accessed April 5, 2025, https://stackoverflow.com/questions/67728919/rendering-three-js-canvas-in-vue-without-using-let-variables-outside-nor-appendc
29. 7 Best Practices for Structuring Large-Scale Vue.js Applications | by Al Emran - Medium, accessed April 5, 2025, https://medium.com/@alemrandev/7-best-practices-for-structuring-large-scale-vue-js-applications-cbf47beedb99
30. Good practices and Design Patterns for Vue Composables - DEV Community, accessed April 5, 2025, https://dev.to/jacobandrewsky/good-practices-and-design-patterns-for-vue-composables-24lk
31. Drie - Vue 3 component library for three.js : r/vuejs - Reddit, accessed April 5, 2025, https://www.reddit.com/r/vuejs/comments/10ikpmq/drie_vue_3_component_library_for_threejs/
32. scottbedard/vue-use-three: An experimental integration of ... - GitHub, accessed April 5, 2025, https://github.com/scottbedard/vue-use-three
33. janvorisek/drie: Drie is a Vue 3 component library for ... - GitHub, accessed April 5, 2025, https://github.com/janvorisek/drie
34. Composables - TresJS, accessed April 5, 2025, https://docs.tresjs.org/api/composables.html
35. 3D Scene with Vue using TresJS - Egghead.io, accessed April 5, 2025, https://egghead.io/tips/Create-a-3D-Scene-with-Vue-using-TresJS~xuoee
36. Create Interactive 3D Experiences with TresJS | egghead.io, accessed April 5, 2025, https://egghead.io/courses/create-interactive-3d-experiences-with-tresjs-00405

7c2
37. Introduction | TresJS, accessed April 5, 2025, https://docs.tresjs.org/guide/
38. TresJS | The solution for 3D on VueJS, accessed April 5, 2025, https://docs.tresjs.org/
39. Load Models - TresJS, accessed April 5, 2025, https://docs.tresjs.org/cookbook/load-models
40. Introduction | TresJS, accessed April 5, 2025, https://tresjs.org/guide/
41. TresJS, a declarative way of creating 3D scenes from Vue components - Vue.js Live 2023, accessed April 5, 2025, https://www.youtube.com/watch?v=AoI9FbJvb2k
42. TresJS - First steps with 3D on Vue - YouTube, accessed April 5, 2025, https://www.youtube.com/watch?v=k2ntrRtR8wc
43. Building a 3D Scene in Nuxt with TresJS | Vue Mastery, accessed April 5, 2025, https://www.vuemastery.com/blog/building-a-3d-scene-in-nuxt-with-tresjs/
44. Getting started with 3D on Vue with TresJS - YouTube, accessed April 5, 2025, https://www.youtube.com/watch?v=ZM7EeS75sYM
45. TresJS - Animate your 3D Objects with Vue - YouTube, accessed April 5, 2025, https://www.youtube.com/watch?v=bDWdikyqfjk
46. Zorger27/Vue-Threejs-1: A web application with a dynamic ... - GitHub, accessed April 5, 2025, https://github.com/Zorger27/Vue-Threejs-1
47. Three.js animate from one position to another - Stack Overflow, accessed April 5, 2025, https://stackoverflow.com/questions/49934594/three-js-animate-from-one-position-to-another
48. Vue animations just got WAY better - YouTube, accessed April 5, 2025, https://www.youtube.com/watch?v=wvINSME-FMs
49. king2088/vue-3d-loader: VueJS and threeJS 3d viewer plugin - GitHub, accessed April 5, 2025, https://github.com/king2088/vue-3d-loader
50. Vue 3D Loader - ThreeJS 3D Viewer, accessed April 5, 2025, https://madewithvuejs.com/vue-3d-loader
51. THREE.js obj not loading in vue - Stack Overflow, accessed April 5, 2025, https://stackoverflow.com/questions/66680497/three-js-obj-not-loading-in-vue
52. 3D visualization techniques based on Three.js and Quasar framework for digital twin systems - SPIE Digital Library, accessed April 5, 2025, https://www.spiedigitallibrary.org/conference-proceedings-of-spie/13239/132390J/3D-visualization-techniques-based-on-Threejs-and-Quasar-framework-for/10.1117/12.3035921.full
53. Using Three.js for 3D Graphics in JavaScript Applications - PixelFreeStudio Blog, accessed April 5, 2025, https://blog.pixelfreestudio.com/using-three-js-for-3d-graphics-in-javascript-applications/
54. State Management in Vue.js: A Complete Guide, accessed April 5, 2025, https://blog.pixelfreestudio.com/state-management-in-vue-js-a-complete-guide/
55. How does Vue.js handle state management in complex applications? - Lemon.io, accessed April 5, 2025,

https://lemon.io/answers/vue-js/how-does-vue-js-handle-state-management-in-complex-applications/
56. What is Vuex? | Vuex, accessed April 5, 2025, https://vuex.vuejs.org/
57. State - Vuex, accessed April 5, 2025, https://vuex.vuejs.org/guide/state
58. Reactive And Simple State Management With Vuex - GeekyAnts Tech Blog, accessed April 5, 2025, https://techblog.geekyants.com/reactive-and-simple-state-management-with-vuex
59. Vue and Threejs - Part One - Stage Right Labs, accessed April 5, 2025, https://stagerightlabs.com/blog/vue-and-threejs-part-one
60. WebGL 3D physics, sound and MIDI in Vue 3 and TypeScript. : r/vuejs - Reddit, accessed April 5, 2025, https://www.reddit.com/r/vuejs/comments/m0ezuo/webgl_3d_physics_sound_and_midi_in_vue_3_and/
61. Different State Management Patterns for VueJS - Daily.dev, accessed April 5, 2025, https://daily.dev/blog/different-state-management-patterns-for-vuejs
62. Is there a recommened VueJS application API and Vuex design pattern - Stack Overflow, accessed April 5, 2025, https://stackoverflow.com/questions/66963930/is-there-a-recommened-vuejs-application-api-and-vuex-design-pattern
63. sdras/three-vue-pattern: A biofeedback visualization made with Three.js, Vue, and LUIS (cognitive services), made with Brian Holt - GitHub, accessed April 5, 2025, https://github.com/sdras/three-vue-pattern
64. State | Pinia, accessed April 5, 2025, https://pinia.vuejs.org/core-concepts/state.html
65. Everything Beyond State Management in Stores with Pinia - GitNation, accessed April 5, 2025, https://gitnation.com/contents/everything-beyond-state-management-in-stores-with-pinia
66. Vue3 Pinia State Management - Stack Overflow, accessed April 5, 2025, https://stackoverflow.com/questions/79309803/vue3-pinia-state-management
67. ebbesand123/vue-d: Client 3D application powered by ... - GitHub, accessed April 5, 2025, https://github.com/ebbesand123/vue-d
68. Pinia | The intuitive store for Vue.js, accessed April 5, 2025, https://pinia.vuejs.org/
69. #53 - Pinia (Vuex 5) State Management - Vue 3 Tutorial - YouTube, accessed April 5, 2025, https://www.youtube.com/watch?v=U_GjNn4i4Ys
70. Master state management in Vue.js with Pinia - Kinsta®, accessed April 5, 2025, https://kinsta.com/blog/vue-pinia/
71. Vue.js State Management: Pinia vs. Vuex - Telerik.com, accessed April 5, 2025, https://www.telerik.com/blogs/vue-js-state-management-pinia-vs-vuex
72. State Management | Vue.js, accessed April 5, 2025, https://vuejs.org/guide/scaling-up/state-management
73. Vue and Threejs - Part Two - Stage Right Labs, accessed April 5, 2025, https://stagerightlabs.com/blog/vue-and-threejs-part-two
74. Post-processing - TresJS, accessed April 5, 2025,

https://post-processing.tresjs.org/guide/
75. TresJS Post-processing v1, accessed April 5, 2025, https://tresjs.org/blog/tresjs-post-processing-v1/
76. How to build the right threejs app architecture? - Questions - three.js forum, accessed April 5, 2025, https://discourse.threejs.org/t/how-to-build-the-right-threejs-app-architecture/69571
77. @tresjs/post-processing - npm, accessed April 5, 2025, https://www.npmjs.com/package/%40tresjs%2Fpost-processing
78. EffectComposer – three.js docs, accessed April 5, 2025, https://threejs.org/docs/examples/en/postprocessing/EffectComposer.html
79. ShaderMaterial – three.js docs, accessed April 5, 2025, https://threejs.org/docs/api/en/materials/ShaderMaterial.html
80. The shader material in threejs and getting started with a little GLSL - Dustin Pfister, accessed April 5, 2025, https://dustinpfister.github.io/2023/01/13/threejs-shader-material/
81. Implement custom WebGL shaders in vuejs - Stack Overflow, accessed April 5, 2025, https://stackoverflow.com/questions/71710027/implement-custom-webgl-shaders-in-vuejs
82. Could Someone help me out with implementing custom shaders? - three.js forum, accessed April 5, 2025, https://discourse.threejs.org/t/could-someone-help-me-out-with-implementing-custom-shaders/65246
83. Modern ThreeJS Shaders with GLSL Vite plugin - YouTube, accessed April 5, 2025, https://m.youtube.com/watch?v=VF2nJxoIA6c&pp=ygULI2dsc2xzaGFkZXI%3D
84. FarazzShaikh/THREE-CustomShaderMaterial: Extend Three.js standard materials with your own shaders! - GitHub, accessed April 5, 2025, https://github.com/FarazzShaikh/THREE-CustomShaderMaterial
85. Performance - Vue.js, accessed April 5, 2025, https://vuejs.org/guide/best-practices/performance
86. Improving the Performance of Vue 3 Applications Using v-memo and KeepAlive, accessed April 5, 2025, https://www.thisdot.co/blog/improving-the-performance-of-vue-3-applications-using-v-memo-and-keepalive
87. How to improve performance on my site? - three.js forum, accessed April 5, 2025, https://discourse.threejs.org/t/how-to-improve-performance-on-my-site/52039
88. How to Optimize Performance in Vue.js Applications: Beginner to Advanced Guide, accessed April 5, 2025, https://dev.to/delia_code/how-to-optimize-performance-in-vuejs-applications-beginner-to-advanced-guide-53db
89. Add an Interactive 3D Model to Your Website // Three.js Tutorial for Beginners - YouTube, accessed April 5, 2025, https://www.youtube.com/watch?v=aOQuuotM-Ww
90. How to Build Immersive-AR Web Applications [Vue.Js and Three.Js ..., accessed

April 5, 2025, https://lightit.io/blog/immersive-ar-web-application/
91. Get correct mouseover interaction in a ThreeJS VueJS app changing the window, accessed April 5, 2025, https://stackoverflow.com/questions/65270851/get-correct-mouseover-interaction-in-a-threejs-vuejs-app-changing-the-window
92. Building an interactive web portfolio with Vue + Three.js — Part One: Introduction and Setup, accessed April 5, 2025, https://medium.com/nicasource/building-an-interactive-web-portfolio-with-vue-three-js-part-one-introduction-and-setup-405426cec84
93. three.js manual, accessed April 5, 2025, https://threejs.org/manual/
94. I am a designer looking for three.js developmont help - Jobs, accessed April 5, 2025, https://discourse.threejs.org/t/i-am-a-designer-looking-for-three-js-developmont-help/47737
95. sascha245/vue-threejs-composer - GitHub, accessed April 5, 2025, https://github.com/sascha245/vue-threejs-composer
96. vue-threejs - NPM, accessed April 5, 2025, https://www.npmjs.com/package/vue-threejs
97. vue-threejs/src/physics/MovementSystem.vue at dev · fritx/vue ..., accessed April 5, 2025, https://github.com/fritx/vue-threejs/blob/dev/src/physics/MovementSystem.vue
98. Compose beautiful scenes with Vue.js and Three.js | by Sascha Braun - Medium, accessed April 5, 2025, https://medium.com/@sascha245/compose-beautiful-scenes-with-vue-js-and-three-js-907f9c0e5eaf
99. How to render threejs AR applications like VR - Stack Overflow, accessed April 5, 2025, https://stackoverflow.com/questions/78999115/how-to-render-threejs-ar-applications-like-vr
100. WebXR API with Vue 3 and Three.js | Playlist - YouTube, accessed April 5, 2025, https://www.youtube.com/watch?v=uLUgdqehB30
101. webXR image tracking: Drag and Drop in AR with ThreeJS - Stack Overflow, accessed April 5, 2025, https://stackoverflow.com/questions/77846718/webxr-image-tracking-drag-and-drop-in-ar-with-threejs
102. 3d gallery with AR support | Webxr api, Three.js and Vue 3 - YouTube, accessed April 5, 2025, https://www.youtube.com/watch?v=9KXz__saMUE
103. Vue VR - Panorama Viewer, accessed April 5, 2025, https://madewithvuejs.com/vue-vr