The "Ethereal Glass & Fluid" Blueprint: Architecting a Site of the Year

1. Executive Vision: Defining "Liquid Intelligence"

The digital landscape of 2025 demands more than functional competence; it requires atmospheric immersion. The objective of this architectural specification is to define the design system and technical stack for a portfolio site capable of securing the "Site of the Year" accolade on Awwwards. The overarching concept, "Liquid Intelligence," represents a fundamental shift in user interface design—moving away from static, page-based metaphors toward a continuous, living environment where content is suspended in a reactive, fluid medium. This document serves as the definitive roadmap for the creative technology and engineering teams, detailing the fusion of interactable fluid dynamics, hyper-realistic glassmorphism, and kinetic typography into a cohesive digital world.

The core philosophy of "Liquid Intelligence" is the juxtaposition of the organic and the crystalline. It explores the tension between the chaotic, uncontrollable nature of fluid dynamics—representing creativity, flow, and the subconscious—and the structured, precise nature of glass—representing clarity, logic, and the editorial voice. By simulating these physical properties in real-time using WebGL, the interface ceases to be a passive container for content and becomes an active participant in the narrative. The user does not simply view the work; they disturb it, ripple through it, and view it refracted through the lens of the agency's distinct perspective.[1]

To achieve this, the technical architecture bypasses traditional DOM-heavy layouts in favor of a hybrid rendering stack. The "Living" Liquid Engine, a custom GPU-accelerated implementation of the Navier-Stokes equations, serves as the background layer, providing a perpetual, non-repeating canvas of motion.[3] Floating above and within this fluid are interface elements rendered in "Glassmorphism 3.0," a technique utilizing advanced physically based rendering (PBR) to simulate chromatic dispersion, thickness, and variable indices of refraction.[5] This creates a sense of deep space, where the "Deep Void" color palette creates an infinite backdrop illuminated by bioluminescent interactions.

This report is exhaustive. It breaks down the physics of the simulation, the mathematics of the optical effects, the intricacies of the typography, and the engineering required to run this simulation at 60 frames per second on consumer hardware. It is not merely a design document; it is a blueprint for building a living digital organism.

---

2. Design System Architecture: The Aesthetic of the

Void

The visual language of the portfolio must command authority while inviting exploration. In 2025, the trend for award-winning digital design has moved toward "Editorial Gravity"—a style that borrows the sophisticated typography and negative space of high-end print magazines but animates them with the unbounded potential of the browser.[7]

2.1 The "Deep Void" Color Theory

Standard dark modes are insufficient for the atmospheric depth required by "Liquid Intelligence." We utilize a "Deep Void" palette, creating a darkness that is not empty but rich and atmospheric, designed to maximize the impact of the fluid simulation's bioluminescence.

The Abyssal Foundation
The primary background is not black (#000000), which flattens the screen, but a "Void Navy" (#0b1020 or #0e0216).8 This hue mimics the light-absorbing properties of deep ocean water or deep space, providing a chromatic context that allows the additive color mixing of the WebGL layer to shine with greater luminance. This dark base is often graded with an "Abyssal Purple" (#1a1033) in the lower frequencies of the fluid simulation, suggesting a space that extends infinitely backward, creating a volumetric stage for the content.9
Bioluminescent Accents
The interaction layer relies on high-energy colors that cut through the darkness. These colors are selected not just for their hue, but for their RGB intensity values, which are deliberately pushed beyond the standard 0-1 range to trigger the bloom post-processing effects.11

- **Liquid Chrome / Cyan:** (#0ff0fc or #5fb7cf) acts as the primary accent for "intelligence"—cursor interactions, active states, and data visualization. It represents the "technological" aspect of the agency's persona.[9]
- **Neon Coral:** (#ff2a6d or #f96cff) provides the "organic" counterpoint. Used for the fluid splats and transition animations, this color vibrates visually against the deep blue background, creating a sense of energy and disruption.[9]
- **Alien Green:** (#b6f486 or #39ffba) is reserved for technical specifications and "system ready" indicators, reinforcing the "Creative Technologist" narrative.[15]

2.2 Editorial Typography: The Voice of the Machine

The typographic strategy rejects the safe neutrality of geometric sans-serifs that dominated the previous decade. The "Site of the Year" aesthetic for 2025 is defined by character, history, and a touch of the uncanny.[7] The typography must feel like it is floating within the fluid, subject to the same physical forces as the graphical elements.

Primary Display: The Dangerous Serif
The headline typeface requires high contrast and sharp, predatory curves. GT Super Display is the primary candidate. Its design, influenced by 1970s phototypesetting, captures the expressive nature of calligraphic motions compelled into stable shapes.18 The razor-sharp

serifs act as visual anchors in the fluid environment, cutting through the softness of the background simulation. Alternatively, Ogg offers a more idiosyncratic, italicized energy, where ligatures and swashes suggest a liquid connection between letters.19 Both fonts possess an "editorial gravity" that elevates the portfolio from a gallery of thumbnails to a curated exhibition.

Secondary Functional: Invisible Precision

To ground the expressive display type, the functional UI—navigation, project metadata, and technical descriptions—utilizes a hyper-legible neo-grotesque like Geist (by Vercel) or Inter.21 These typefaces are chosen for their neutrality and high readability at small sizes. They represent the "container" or the "glass" holding the content—transparent, structured, and precise.

2.3 Layout Philosophy: The Fluid Grid

The layout follows a "Fluid Grid" philosophy. While the DOM elements are anchored to a responsive 12-column grid for accessibility and SEO, the 3D WebGL layer operates on a continuous, normalized coordinate system.

- **Z-Index Stratification:** The interface is conceived as a deep stack. Layer 0 is the Fluid Engine (Infinite depth). Layer 1 contains the 3D Glass Objects (The content vessels). Layer 2 holds the floating Typography (The narrative).
- **Glass Containers:** Instead of solid cards, content is housed within MeshTransmissionMaterial planes. These glass panels refract the fluid background, meaning the background of a text block is not a static color, but a distorted, moving view of the simulation behind it. This creates a dynamic dependency between the UI and the environment.[5]

---

3. The "Living" Liquid Engine: Physics & Simulation

The "Living" Liquid Engine is the heartbeat of the experience. Unlike pre-rendered video loops or simple noise shaders, this engine solves the Navier-Stokes equations for incompressible flow in real-time on the GPU. This ensures that the background is never static; it reacts to the user's presence, retaining the memory of their movements in swirling eddies that dissipate slowly over time.

3.1 The Mathematical Foundation: Navier-Stokes

The simulation is built upon the grid-based (Eulerian) approach to fluid dynamics, which is computationally superior for the continuous, screen-filling effects required here compared to particle-based (Lagrangian) methods.[23] The fluid state is described by a velocity field $\mathbf{u}$ and a density (color) field ρ, which evolve according to the incompressible Navier-Stokes equations [4]:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla) \mathbf{u} - \nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{F}$$
The components of this equation dictate the behavior of our "Liquid Intelligence":

1. **Advection** ($-(\mathbf{u} \cdot \nabla) \mathbf{u}$): The transport of the fluid by its own velocity. This creates the swirling motion.
2. **Pressure** ($-\nabla p$): The internal force that keeps the fluid from compressing or expanding, maintaining volume.
3. **Diffusion** ($\nu \nabla^2 \mathbf{u}$): The viscosity or thickness of the fluid.
4. **External Forces** ($\mathbf{F}$): The input from the user (mouse, scroll) or environment (gravity, wind).

3.2 The Simulation Pipeline: FBO Ping-Pong

To solve these equations in WebGL, we utilize a technique known as "Ping-Ponging" with Frame Buffer Objects (FBOs). Since a shader cannot read from and write to the same texture simultaneously, we maintain two textures for every fluid property (velocity and density): a read texture and a write texture. In each frame, the simulation steps are executed by rendering full-screen quads, reading from the current state, and writing the new state to the destination FBO, then swapping the textures.[3]

Step 1: Advection (The Motion)

The advection step is responsible for moving the fluid density and velocity along the velocity field. We utilize the **Semi-Lagrangian** scheme for unconditional stability. Instead of calculating where a fluid parcel *goes* (which can land between grid cells and cause instability), we calculate where the fluid at the current pixel *came from*.[26]

- **Mechanism:** For each fragment at position $\mathbf{x}$, we trace the velocity vector backwards in time: $\mathbf{x}_{prev} = \mathbf{x} - \mathbf{u}(\mathbf{x}) \delta t$. We then sample the fluid quantity (color or velocity) at $\mathbf{x}_{prev}$ using linear interpolation.[3]
- **Dissipation:** We explicitly disable the viscous diffusion solver for the velocity field. The linear interpolation inherent in the texture sampling provides a small amount of "numerical dissipation," which acts as a virtual viscosity. This is a common optimization in real-time graphics that saves expensive diffusion iterations while preventing the fluid from becoming too chaotic.[3]

Step 2: External Forces (The Interaction)

This is where the user enters the simulation. We inject forces based on the mouse or touch input.

- **Gaussian Splat:** When the input device moves, we calculate the delta velocity. We render a "splat" into the velocity and density FBOs at the cursor coordinates. This splat uses a Gaussian distribution function (e^{-r^2}) to create a soft, organic impulse rather than a

hard circle.[4]

- **Color Injection:** Simultaneously, we inject the "Neon Coral" or "Liquid Chrome" dye into the density field. The intensity of this color is modulated by the speed of the mouse movement—faster movements create brighter, more turbulent splashes.[25]

Step 3: Divergence (The Math of Incompressibility)

Real water does not compress. If fluid flows into a grid cell, an equal amount must flow out. The advection step often violates this, creating "divergence."

- Calculation: We calculate the divergence of the velocity field for each cell by comparing the velocities of its neighbors:

  $$div(\mathbf{x}) = \frac{1}{2\epsilon} ((u_{right} - u_{left}) + (v_{top} - v_{bottom}))$$

  This scalar value tells us how much "excess" fluid is in each cell.4

Step 4: Pressure Solve (Jacobi Iteration)

To correct the divergence, we must calculate a pressure field that opposes it. This requires solving the Poisson equation, which is a system of linear equations.

- Jacobi Iteration: We solve this iteratively on the GPU. For each iteration, a cell's pressure is updated based on the average pressure of its neighbors and the divergence calculated in the previous step.

  $$p_{new} = \frac{p_{left} + p_{right} + p_{bottom} + p_{top} - \alpha \cdot div}{\beta}$$
- **Performance vs. Accuracy:** A true solution might take hundreds of iterations. For our visual purposes, **20 to 40 iterations** are sufficient to create visually incompressible flow without destroying the frame rate.[28] This is the most computationally expensive part of the pipeline.

Step 5: Gradient Subtraction

Finally, we subtract the gradient of the computed pressure field from the advected velocity field. The result is a **divergence-free** velocity field that respects the physics of mass conservation. This new field is used as the input for the next frame.[4]

3.3 Vorticity Confinement: Restoring the Swirl

Numerical dissipation in the advection step tends to dampen small-scale swirls (vorticity), making the fluid look like thick syrup or smoke. To achieve the "Ethereal" look, we apply **Vorticity Confinement**.

- **Technique:** We calculate the "curl" of the velocity field to identify where the fluid is spinning. We then apply a boosting force in the direction of the curl, reinjecting energy into the vortices.

- **Visual Result:** This maintains the intricate, whiskey-smoke-like tendrils that make the simulation feel alive and detailed, rather than blurry and sluggish.[4]

---

4. Glassmorphism 3.0: Advanced Material Science

The "Glass" in this blueprint is not merely a CSS backdrop-filter: blur(). It is a fully volumetric, ray-marched material simulation that interacts with the 3D lighting environment and refracts the fluid engine behind it. This is "Glassmorphism 3.0"—characterized by accurate refraction, chromatic aberration, and thickness simulation.

4.1 The MeshTransmissionMaterial

The cornerstone of this effect is the MeshTransmissionMaterial from the @react-three/drei library. This material extends the standard THREE.MeshPhysicalMaterial to provide physically plausible transmission properties without the extreme performance cost of real-time ray tracing.[5]

Key Configuration for Hyper-Realism

To achieve the "Site of the Year" look, the material parameters must be tuned to simulate a dense, crystal-like medium rather than thin window glass.

Parameter	Recommended Value	Visual Impact
Transmission	1.0	Allows light to pass fully through the object.
Thickness	2.0 - 5.0	Simulates a solid volume. Higher values increase the refraction distance, causing the background fluid to warp significantly, acting as a lens.[5]
Roughness	0.15	A purely smooth surface (0.0) looks like plastic. A slight roughness catches the specular highlights from the studio lighting, defining the object's geometry.

IOR	1.5 - 1.7	The Index of Refraction controls the bending of light. We can dynamically animate this value on scroll to create a "breathing" effect where the glass density appears to change.
Chromatic Aberration	0.06 - 0.15	**Crucial.** This simulates the dispersion of light (like a prism), splitting the image behind the glass into RGB channels at the edges. This adds a cinematic, "imperfect" lens quality that signifies high production value.[5]
Anisotropy	0.5	Stretches the specular highlights along the tangent, giving the glass a "brushed" or "machined" appearance, enhancing the kinetic feel during rotation.[5]

4.2 The Transmission Sampler Challenge

A significant technical challenge in WebGL refraction is the "Buffer Problem." For a material to refract the scene, it must access a texture of what lies behind it.

- **The Issue:** The default transmissionSampler generates a buffer of the opaque scene objects. However, it often fails to "see" other transparent objects. If we stack multiple glass planes (e.g., a glass menu over a glass hero image), the front glass might render the back glass as invisible or black.[5]
- **The Solution:** We must manage the render order and potentially use custom FBOs. For complex overlapping glass, we can disable the automatic transmissionSampler and manually pass a rendered texture of the background (Fluid Engine) to the background prop of the material. This ensures that the glass refracts the fluid correctly, even if it cannot perfectly refract other glass objects.[5] Alternatively, setting renderPriority helps the renderer sort the draw calls correctly, drawing the back glass before the front glass.[33]

4.3 Backside Rendering & Volume

Standard WebGL materials cull back-facing triangles. For a thick glass object like a torus or

sphere, this destroys the illusion of volume.

- **Backside Prop:** We explicitly enable backside={true}. This forces the shader to calculate the refraction for the back surface of the mesh as well as the front.
- **Beer's Law:** By rendering the backside, the material can approximate Beer's Law (absorption), where the glass gets darker or more tinted the thicker it is. This adds "weight" to the digital objects.[5]

---

5. The Hybrid Render Stack: WebGL + DOM

The "Ethereal Glass & Fluid" blueprint creates a unified world, but technically, it spans two distinct rendering contexts: the HTML DOM (for accessible text and layout) and the WebGL Canvas (for fluid and glass). Merging these seamlessly is the primary UX challenge.

5.1 The "Sandwich" Architecture

The application is structured as a visual sandwich:

1. **Background (Layer 0):** The Canvas element containing the R3F scene.
    - *Fluid Engine Mesh:* A full-screen plane at z = -10.
    - *3D Glass Objects:* Floating meshes at z = 0 to z = 5.
2. **Foreground (Layer 1):** The HTML DOM overlay.
    - *Typography:* Headers, paragraphs, navigation.
    - *Interactions:* Buttons, links.

5.2 Synchronized Scroll Journey: Lenis + R3F

Standard browser scrolling is discrete and often jagged. It does not communicate well with the 60fps continuous loop of the WebGL engine. To unify the "Scroll Journey," we employ **Lenis**, a specialized smooth-scrolling library.[34]

- **Velocity Sync:** Lenis provides precise data on scroll velocity (lenis.velocity). We feed this value directly into the WebGL uniforms.
    - *Effect:* When the user scrolls quickly, the fluid simulation receives a "turbulence" boost, causing the background to ripple in the direction of the scroll.
    - *Effect:* The glass objects can elongate or distort (vertex shader manipulation) based on scroll speed, creating a "warp speed" effect.
- **ScrollControls:** We use @react-three/drei's ScrollControls to map the HTML scroll progress (0 to 1) to the 3D camera's position on a spline path. This allows for a cinematic camera rig that pans, tilts, and zooms to focus on specific 3D artworks as the user reads the corresponding DOM text.[35]

5.3 The Refraction Paradox: DOM vs. WebGL

A critical limitation is that WebGL refraction *cannot* physically bend DOM elements (like an

<h1> tag) because the DOM sits *on top* of the Canvas.

- **The Illusion:** To achieve the effect of glass refracting text, we must render the text *inside* the WebGL scene.
- **Hybrid Text Strategy:**
  1. **Hero/Artistic Text:** We use **Troika-Three-Text** to render the large, "Liquid Intelligence" display typography as 3D meshes inside the scene.[37] These are placed *behind* the glass objects, allowing them to be fully refracted, distorted, and colored by the glass material.[37]
  2. **Functional/SEO Text:** Paragraphs and navigation remain in the DOM for accessibility. We use the <Html> component from Drei with occlude="blending". This allows 3D glass objects to visually pass "in front" of the DOM text (by hiding the text when occluded), maintaining the depth illusion without sacrificing usability.[38]

---

6. Lighting, Atmosphere & Post-Processing

The difference between a "tech demo" and a "Site of the Year" lies in the lighting and compositing. We treat the browser window as a cinematography set.

6.1 Cinematic Lighting Setup

Glass requires an environment to reflect. Without an environment map, glass looks invisible.

- **HDRI Studio:** We utilize a custom High Dynamic Range Image (HDRI) acting as a "studio" environment. This HDRI should feature sharp, high-contrast softbox lights on a dark background. This ensures that the glass edges catch bright, white specular highlights, defining their silhouette against the dark fluid.[32]
- **Volumetric Spotlights:** We place SpotLight objects with volumetric properties (using GodRay effects or simple cone geometry with gradient opacity) to illuminate the floating glass artifacts. These lights should cast subtle shadows onto the "fluid" floor, grounding the objects in the space.

6.2 Selective Bloom & Tone Mapping

To achieve the "Out of This World" bioluminescence, we implement a selective bloom pipeline using @react-three/postprocessing.

- **Luminance Threshold:** We set the bloom luminanceThreshold to 1.0. This means standard colors (0.0 to 1.0) will *not* glow.
- **HDR Colors:** We define our "Neon Coral" and "Liquid Chrome" colors using RGB values well above 1.0 (e.g., [4.0, 0.2, 2.5]). Because these values exceed the threshold, the bloom pass isolates them and applies the glow, while keeping the rest of the interface crisp and readable.[11]
- **Tone Mapping Disable:** Crucially, we must set toneMapped={false} on the materials of

the glowing objects. Standard tone mapping (like ACESFilmic) compresses high-dynamic-range values back down to 0-1, which would kill the bloom effect. Disabling it allows the raw, intense color values to hit the post-processing stack.[11]

6.3 Grain & Noise

A digital simulation can look too perfect and sterile. To unify the fluid, glass, and text layers, we apply a subtle **Noise/Grain** pass at the very end of the effect composer chain. This adds a filmic texture ("dithering") that reduces color banding in the dark gradients and gives the experience a tactile, analog quality.[5]

---

7. Performance Engineering & Optimization

Ideally, this site runs at 60fps on a MacBook Air, not just a dedicated GPU workstation. Optimization is an engineering requirement, not an afterthought.

7.1 Dynamic Resolution Scaling (DPR)

We utilize the PerformanceMonitor component from Drei. This creates a feedback loop: if the frame rate drops below a threshold (e.g., 50fps), the system automatically lowers the dpr (Device Pixel Ratio) of the canvas.

- **Impact:** A fluid simulation rendered at 0.5x resolution and upscaled looks "softer" but is still visually acceptable, whereas a UI running at 20fps creates input lag and frustration. This adaptive scaling ensures the site remains fluid on lower-end devices.[41]

7.2 Simulation Grid Resolution

We decouple the fluid simulation grid size from the screen resolution.

- **Optimization:** A 1080p screen does not need a 1920x1080 fluid grid. A grid of **256x256** or **512x512** is often sufficient. We stretch this smaller texture to fill the screen using linear interpolation (OES_texture_float_linear). This reduces the number of texels processed by the Navier-Stokes shaders by over 90%, massively freeing up GPU resources for the glass refraction.[25]

7.3 Asset Optimization

- **Draco Compression:** All 3D geometry (glass shapes) is compressed using Draco. This reduces the geometry payload by ~80%, improving the First Contentful Paint (FCP).[42]
- **Textureless Design:** The aesthetic strategy intentionally avoids heavy diffuse textures (JPG/PNG). The visual detail comes from *computation* (shaders, refraction, fluid), which is bandwidth-light, rather than *transmission* (downloading large images).

7.4 Shader Warm-Up

Complex shader pipelines (Fluid + Transmission + Bloom) can cause a noticeable stutter (jank) on the very first frame as the GPU compiles the GLSL programs.

- **Strategy:** During the preloader sequence (while the "Loading..." percentage is visible), we force the renderer to render the main scene off-screen for a few frames with 0 opacity. This forces the driver to compile the shaders and upload the textures before the user sees the content, ensuring the "curtain drop" reveal is buttery smooth.

---

8. Conclusion

The "Ethereal Glass & Fluid" blueprint represents a convergence of high-concept art direction and rigorous graphics engineering. It rejects the static nature of the traditional web in favor of a "living" system. By implementing a physically accurate Navier-Stokes fluid engine, harnessing the optical complexity of Glassmorphism 3.0, and synchronizing these elements with a kinetic scroll journey, we create a digital artifact that demands attention.

This is not a portfolio that is simply "read." It is a world that is inhabited. The user's cursor becomes a brush, the scrollbar a throttle, and the screen a window into a Deep Void of Liquid Intelligence. This holistic integration of aesthetic vision and technical execution is the precise formula required to win "Site of the Year."

---

9. Appendix: Technical Reference Tables

Table 1: Fluid Engine Shader Parameters

Parameter	Value	Description
Grid Size	256 x 256 (Mobile: 128x128)	Resolution of the simulation FBOs.
Delta Time (δt)	0.016 (Fixed)	Time step for physics. Fixed step preferred for stability.
Viscosity	0.0 (Implicit)	Relies on linear interpolation for dissipation.

Pressure Iterations	20	Jacobi solver loops per frame.
Splat Radius	0.025 (Normalized)	Size of the Gaussian impulse from cursor.
Curl Strength	25.0	Vorticity confinement coefficient.
Dissipation Rate	0.98	How quickly the color/velocity fades (0-1).

Table 2: Glass Material Configuration (MeshTransmissionMaterial)

Prop	Value	Effect
transmission	1.0	Full transparency.
thickness	3.5	Strong refractive distortion (lens).
roughness	0.15	Slight surface gloss definition.
chromaticAberration	0.08	Prismatic color splitting.
anisotropy	0.5	Brushed/Kinetic specular highlights.
distortion	0.2	Subtle surface noise warping.
backside	true	Volumetric rendering (Beer's law).
resolution	512	Resolution of the transmission buffer.

Table 3: Post-Processing & Bloom Settings

Effect	Prop	Value	Purpose
Bloom	luminanceThreshold	1.0	Only glow colors > 1.0 (HDR).
Bloom	mipmapBlur	true	Soft, atmospheric glow scatter.
Bloom	intensity	1.5	Strength of the bioluminescence.
Noise	opacity	0.05	Film grain for texture.
Vignette	darkness	0.5	Focuses attention on center.

Works cited

1. Sites Of The Day - Awwwards, accessed January 1, 2026, https://www.awwwards.com/websites/sites_of_the_day/
2. LIQUID DESIGN Ltd. - Awwwards Nominee, accessed January 1, 2026, https://www.awwwards.com/sites/liquid-design-ltd
3. WebGL Fluid Simulation - 29a.ch, accessed January 1, 2026, https://29a.ch/2012/12/16/webgl-fluid-simulation
4. Chapter 38. Fast Fluid Dynamics Simulation on the GPU - NVIDIA Developer, accessed January 1, 2026, https://developer.nvidia.com/gpugems/gpugems/part-vi-beyond-triangles/chapter-38-fast-fluid-dynamics-simulation-gpu
5. MeshTransmissionMaterial - React Three Drei, accessed January 1, 2026, https://drei.docs.pmnd.rs/shaders/mesh-transmission-material
6. Shining a light on Caustics with Shaders and React Three Fiber - Maxime Heckel Blog, accessed January 1, 2026, https://blog.maximeheckel.com/posts/caustics-in-webgl/
7. 50 fonts that will be popular with designers in 2025 | Creative Boom, accessed January 1, 2026, https://www.creativeboom.com/resources/top-50-fonts-in-2025/
8. Call of the Void Color Palette, accessed January 1, 2026, https://www.color-hex.com/color-palette/19079

9. Top 15 Futuristic Color Palettes for Creative Projects With HEX Codes - Filmora, accessed January 1, 2026, https://filmora.wondershare.com/video-creative-tips/futuristic-color-palette.html
10. The Void Color Scheme - Palettes - SchemeColor.com, accessed January 1, 2026, https://www.schemecolor.com/the-void.php
11. Bloom - React Postprocessing, accessed January 1, 2026, https://react-postprocessing.docs.pmnd.rs/effects/bloom
12. Three Fibre: BLOOM - How to set different intensity for different objects? : r/threejs - Reddit, accessed January 1, 2026, https://www.reddit.com/r/threejs/comments/1ei5q98/three_fibre_bloom_how_to_set_different_intensity/
13. Futuristic Color Palette, accessed January 1, 2026, https://www.color-hex.com/color-palette/95007
14. 25+ Aesthetic Color Palettes, for Every Aesthetic (with Hex Color Codes) - Gridfiti, accessed January 1, 2026, https://gridfiti.com/aesthetic-color-palettes/
15. ALIENS Color Palette, accessed January 1, 2026, https://www.color-hex.com/color-palette/16419
16. 10 FREE Color Palettes Inspired by Sci-Fi Classics - Shutterstock, accessed January 1, 2026, https://www.shutterstock.com/blog/sci-fi-color-palettes
17. The Best Wedding Font Combinations for a Timeless, Elegant Celebration - Avelã White, accessed January 1, 2026, https://www.avelawhite.com/journal/our-favourite-fonts-of-2025
18. GT Super – Typeface Specimen and License Purchase - Grilli Type, accessed January 1, 2026, https://www.grillitype.com/typeface/gt-super
19. Ogg Font Combinations & Free Alternatives - Typewolf, accessed January 1, 2026, https://www.typewolf.com/ogg
20. Ogg Font Pairings & Alternatives - MaxiBestOf, accessed January 1, 2026, https://maxibestof.one/typefaces/ogg
21. 100 Best Free Fonts for Designers in 2025 - Awwwards, accessed January 1, 2026, https://www.awwwards.com/best-free-fonts.html
22. Inter font pairing with GT Super - MaxiBestOf, accessed January 1, 2026, https://maxibestof.one/typefaces/inter/pairing/gt-super
23. amandaghassaei/FluidSimulation: WebGL shader for mixed grid-particle fluid simulation, accessed January 1, 2026, https://github.com/amandaghassaei/FluidSimulation
24. Building a Real-Time Fluid Simulator on the GPU (CUDA + OpenGL) - Medium, accessed January 1, 2026, https://medium.com/@noureddach/building-a-real-time-fluid-simulator-on-the-gpu-cuda-opengl-fluid-simulation-part-5-8fa33c27fc22
25. How to create a cursor animation like in Lusion.co WebGL (three.js) : r/threejs - Reddit, accessed January 1, 2026, https://www.reddit.com/r/threejs/comments/1fzr6vf/how_to_create_a_cursor_animation_like_in_lusionco/
26. Navier-Stokes & Flow Simulation, accessed January 1, 2026, https://www.cs.rpi.edu/~cutler/classes/advancedgraphics/S17/lectures/07_fluid_si

mulation.pdf
27. Navier-Stokes - Explanations, accessed January 1, 2026, https://piellardj.github.io/navier-stokes-webgl/readme/
28. Bixio999/3d-fluid-simulation: A real-time 3D fluid simulation in OpenGL - GitHub, accessed January 1, 2026, https://github.com/Bixio999/3d-fluid-simulation
29. Simple Fluid Simulation at The Little Grasshopper, accessed January 1, 2026, https://prideout.net/blog/old/blog/index.html@p=58.html
30. ENHANCED DETAIL FLUID SIMULATION USING IMPLICIT DENSITY-INVARIANCE ALGORITHM - DigiPen, accessed January 1, 2026, https://www.digipen.edu/sites/default/files/public/docs/theses/shuo-wei-chang-digipen-master-of-science-in-computer-science-thesis-enhanced-detail-fluid-simulation-using-implicit-density-invariance-algorithm.pdf
31. Playing with Light and Refraction in Three.js: Warping 3D Text Inside a Glass Torus, accessed January 1, 2026, https://tympanus.net/codrops/2025/03/13/warping-3d-text-inside-a-glass-torus/
32. How to Make a 3D Glass Effect using Three.js and Next.js - YouTube, accessed January 1, 2026, https://www.youtube.com/watch?v=9FDt6tuFP-k
33. Problems with
34. 14islands/r3f-scroll-rig: A react-three-fiber scroll-rig for syncing 3D meshes and DOM elements. - GitHub, accessed January 1, 2026, https://github.com/14islands/r3f-scroll-rig
35. React Three Fiber tutorial - Scroll Animations - YouTube, accessed January 1, 2026, https://www.youtube.com/watch?v=pXpckHDDNYo
36. How to change camera position on scroll in react-three-fibre? - Questions, accessed January 1, 2026, https://discourse.threejs.org/t/how-to-change-camera-position-on-scroll-in-react-three-fibre/42771
37. How to create glass material that refracts elements in DOM? - Questions - three.js forum, accessed January 1, 2026, https://discourse.threejs.org/t/how-to-create-glass-material-that-refracts-elements-in-dom/53625
38. How to add dom elements behind the three.js model in canvas? - Stack Overflow, accessed January 1, 2026, https://stackoverflow.com/questions/79325358/how-to-add-dom-elements-behind-the-three-js-model-in-canvas
39. Inserting content (text, videos, etc..) in Canvas, behind a model - Questions - three.js forum, accessed January 1, 2026, https://discourse.threejs.org/t/inserting-content-text-videos-etc-in-canvas-behind-a-model/61254
40. Scroll, Refraction and Shader Effects in Three.js and React - Codrops, accessed January 1, 2026, https://tympanus.net/codrops/2019/12/16/scroll-refraction-and-shader-effects-in-three-js-and-react/
41. Scaling performance - React Three Fiber, accessed January 1, 2026, https://r3f.docs.pmnd.rs/advanced/scaling-performance

42. How to Make a 3D Glass Effect using Three.js and React - Olivier Larose's Blog, accessed January 1, 2026, https://blog.olivierlarose.com/tutorials/3d-glass-effect
43. Threejs & React Avatar Builder Tutorial - Part 7: Post Processing - YouTube, accessed January 1, 2026, https://www.youtube.com/watch?v=wRd9PQ8MD9A