Architectural Specification and Implementation of Real-Time Atmospheric Laser Rig Simulation and Non-Preemptive Web Audio Pipeline

Operating System Audio Focus Architecture and Non-Preemptive Web Media Handling

Mechanism of Android Audio Focus Preemption

Mobile operating systems enforce strict media concurrency models to prevent competing applications from outputting conflicting acoustic streams simultaneously. Within Android, this arbitration is governed by the native AudioManager service through an explicit state-machine token system known as Audio Focus. When a standard HTML5 <audio> or <video> element initializes or initiates playback within a browser engine such as Chromium (Blink), the browser's media pipeline instantiates an internal MediaSessionImpl bridge. If the media resource contains an active audio track and is unmuted, Blink registers the playback instance with the Android operating system by requesting an exclusive audio focus lock through the native system call:
\text{Request Focus: }
\mathtt{AudioManager.requestAudioFocus}(\mathtt{AUDIOFOCUS_GAIN})
This system call instructs the Android media server that the requesting application requires full, persistent audio primacy. Consequently, the operating system dispatches an asynchronous AUDIOFOCUS_LOSS notification to all other active media services. Background audio daemons, including YouTube Music, Spotify, and podcast players, are architected to immediately halt decoding and transition into a paused state upon receiving AUDIOFOCUS_LOSS to avoid chaotic acoustic overlap.
Chromium on Android couples audio focus acquisition with the system-wide Media Notification Service and remote playback infrastructure. When an unmuted media element begins buffering or transitions into an active playback lifecycle, Blink exposes media transport controls to the Android notification shade and lock screen via the system Universal Media Object (UMO). If a web application instantiates an <audio> tag on page load—even if the element is ostensibly waiting for user input—browser pre-caching behaviors and media engine hydration routines frequently register the stream prematurely, pre-allocating platform audio tracks and preempting external background audio without explicit end-user intent.

Audio Pipeline Configuration Matrix

To eliminate background audio termination, web media elements must be decoupled from the system's exclusive media control bus. The following matrix delineates browser audio configurations, their underlying Android OS directives, and their subsequent impact on concurrent external media playback.

Configuration Strategy	Browser API / Attribute Directive	Native Android Focus Directive	System UMO / Lock Screen Exposure	Background Player Impact (e.g., YouTube Music)
Default HTML5 <audio>	Unconfigured <audio src="..." autoplay>	AUDIOFOCUS_GAIN	Exposed; displays track controls and scrub bar	**Terminated**: Dispatches permanent AUDIOFOCUS_[span_1](start_span)[span_1](end_span)LOSS
Hydration-Preloaded	<audio preload="auto"> or <audio prelo[span_5](start_span)[span_5](end_span)[span_9](start_span)[span_9](end_span)ad="metadata">	AUDIOFOCUS_GAIN_TRANSIENT during probe	Transiently bound during media buffer initialization	**Interrupted**: Background stream pauses or glitches during tab load
	Decoupled User-Triggered	preload="none", disableRemotePlayback=true	Deferred; suppressed until explicit DOM click event	Completely suppressed; no lock screen remote registered
W3C Audio Session API	navigator.audioSession.type = "ambient"	AUDIOFOCUS_NONE / Mixed low-priority stream	Suppressed from media routing infrastructure	**Preserved**: Audio mixes concurrently without ducking or pausing
Web Audio Graph Mixing	Native AudioContext with zero-gain ducking bypass	Internal audio synthesis routing without Blink UMO bind	Suppressed; bypasses HTMLMediaElement lockscreen pipeline	**Preserved**: Complete multi-application audio coexistence

Decoupling and Non-Preemptive Integration Architecture

Mitigating the audio hijack requires a multi-tiered defense targeting HTML markup, the DOM lifecycle, the Media Session API, and hardware audio session routing. The initial vulnerability arises during DOM construction when browsers aggressively scan attributes to optimize media playback pipelines. Declaring preload="none" prevents the browser engine from initiating network sockets, pre-buffering chunk headers, or initializing OS-level audio decoders upon page hydration. Media sources must instead be assigned programmatically strictly within an explicit user activation handler, such as a physical tap on an activation control.

Concurrently, the media element must be isolated from mobile casting and operating system remote controls. Setting the HTML attribute disableremoteplayback and the corresponding DOM property HTMLMediaElement.disableRemotePlayback = true strips the element of its ability to query or connect to external casting devices, such as Chromecast and Android MediaRouter endpoints. This prevents Blink from escalating the media instance to an operating system-level interactive media session.

To complete the decoupling, the web application must neutralize the Media Session coordinator. In Chromium, the navigator.mediaSession interface coordinates playback state with the Android SystemUI notification controller. By setting navigator.mediaSession.playbackState = 'none' and nullifying all media action handlers, the application instructs the browser to tear down its lock-screen transport notification, preventing Blink from renewing AUDIOFOCUS_GAIN while backgrounded. Where supported by modern mobile engines, the W3C Audio Session API (navigator.audioSession) enables web clients to inform the host platform of their acoustic profile. Setting navigator.audioSession.type = "ambient" explicitly classifies the page's acoustic output as incidental background ambiance. Unlike the default "playback" type, which mandates exclusive platform access and pauses competing audio, the "ambi[span_37](start_span)[span_37](end_span)[span_40](start_span)[span_40](end_span)ent " flag mandates that the operating system mix the application's digital audio output directly with active background streams without ducking or eviction.

Photometric Modeling and Multi-Pass Laser Beam Geometry

Additive Photon Blending Mechanics

Standard 2D vector rendering algorithms compute color values using standard alpha compositing based on the Porter-Duff Source Over mode:
$$C_{out} = C_{src} \alpha_{src} + C_{dst} \alpha_{dst} (1 - \alpha_{src})$$
While suitable for opaque geometric illustrations, Source Over fails to replicate the physical behavior of coherent monochromatic light passing through scattering media. Photons traversing an optical path do not physically occlude one another; rather, radiant flux is additive across the line of sight.
In HTML5 Canvas 2D contexts, physical radiance summation is achieved by setting the compositing operation to additive mode:
$$\mathtt{ctx.globalCompositeOperation} = \text{'lighter'}$$
This operation configures the rasterization pipeline to execute pure additive blending:
$$C_{out} = \min(255, \, C_{src} + C_{dst}) \alpha_{out} = \min(1, \, \alpha_{src} + \alpha_{dst})$$
Under additive compositing, chromatic channels sum directly across intersecting paths. For instance, when a monochromatic 532 nm DPSS green vector (0, 255, 0) intersects a 638 nm diode red vector (255, 0, 0), the red and green color channels combine at the point of intersection to produce a saturated yellow node (255, 255, 0). As additional volumetric passes, ambient scattering envelopes, and specular reflections converge upon the same pixel coordinates, individual color components saturate toward full white (255, 255, 255). This mechanism accurately reproduces the optical blooming and sensor overexposure observed in high-intensity concert laser installations.

Optical Scattering Physics in Particulate Atmosphere

The visibility of a laser beam through air depends on the suspension of micron-scale liquid droplets, such as glycol or mineral oil-based theatrical fog. As light propagates along a vector $\vec{r}$, its radiance attenuates according to the Beer-Lambert extinction law:
$$I(s) = I_0 \exp\left(-\int_{0}^{s} \beta_{ext}(x) \, dx \right)$$

where \beta_{ext} = \beta_{scat} + \beta_{abs} represents the total extinction coefficient. Because theatrical smoke particles exhibit radii (r \approx 0.5\text{--}5\,\mu\text{m}) comparable to or larger than optical wavelengths (\lambda \approx 400\text{--}700\text{ nm}), scattering operates predominantly within the Mie scattering regime rather than molecular Rayleigh scattering. The angular distribution of scattered intensity I_{scat}(\theta) is highly anisotropic, governed by the Henyey-Greenstein phase function:
p(\theta) = \frac{1}{4\pi} \frac{1 - g^2}{\left(1 + g^2 - 2g \cos\theta\right)^{3/2}}
where g \in (0.7, 0.95) is the forward-scattering asymmetry parameter.
When observing a stage laser rig, light scattered forward toward the audience or camera creates an intense volumetric sheath surrounding the coherent beam axis. Simulating this phenomenon on a discrete 2D raster surface requires a multi-tier optical model that separates the total radiant profile into distinct scattering components: a broad, low-frequency volumetric forward-scatter envelope, an intermediate spectral divergence channel, and a high-frequency, non-scattered paraxial core.

Triple-Pass Beam Geometry Architecture

To simulate a concert-grade solid-state laser rig mounted to an automotive chassis or roof rack, the vector rendering engine executes a sequential triple-pass drawing model along each calculated optical vector:
The first layer is the Outer Volumetric Haze Pass, which models forward Mie scattering throughout the surrounding fog boundary layer. This pass is rendered with an expanded stroke diameter of approximately 20\text{ px}, low alpha saturation between 15\% and 20\%, and a diffuse Gaussian blur filter (\sigma \approx 18\text{--}20\text{ px}). This raises the local black level of the canvas and creates an expansive, glowing scattering envelope.
The second layer is the Mid Spectral Beam Pass, which simulates the visible collimated beam waist and laser divergence sheath. Rendered with a stroke width of 6\text{ px} using fully saturated monochromatic primaries (such as #00ff44 for DPSS green or #ff1133 for high-output red diode arrays), this pass defines the crisp vector boundary and provides the dominant perceptual chromaticity.
The third layer is the Piercing Hot Core Pass, which simulates paraxial photon density where power flux exceeds the dynamic range of human vision and camera sensors. Rendered with an ultra-fine stroke diameter of 1.5\text{ px} using pure achromatic white (#ffffff), this path sits precisely on the central vector axis over the mid-spectral layer under additive mode. It visually fuses with the chromatic layers beneath it, creating an intensely sharp, burning white core surrounded by a saturated spectral corona.

Multi-Pass Laser Rendering Parameter Matrix

Geometric Pass Layer	Physical Optical Counterpart	Stroke Width	Target Color Profile	Blur Filter Radius	Composite Alpha	Canvas Composite Mode
Pass 1: Outer Haze	Atmospheric Mie forward scatter	18.0\text{--}24.0\text{ px}	Highly saturated base hue	15\text{--}20\text{ px}	0.15\text{--}0.20	lighter
Pass 2: Mid Spectral	Collimated beam waist /	5.0\text{--}6.5\text{ px}	Pure neon primary /	None (0\text{ px})	0.85\text{--}1.00	lighter

Geometric Pass Layer	Physical Optical Counterpart	Stroke Width	Target Color Profile	Blur Filter Radius	Composite Alpha	Canvas Composite Mode
	divergence		secondary			
Pass 3: Hot Core	Saturated paraxial flux axis	1.2\text{--}1.8 \text{ px}	Pure Achromatic White (#ffffff)	None (0\text{ px})	1.00	lighter

ILDA Galvanometer Kinematics and Aerodynamic Projection Geometry

Galvanometer Electromechanical Kinematics

Concert laser projectors do not emit static lines; their beams are steered dynamically across two orthogonal axes (X and Y) using closed-loop moving-magnet galvanometer scanners. An International Laser Display Association (ILDA) standard scanner consists of an ultra-low-inertia rotor, a high-reflectivity dielectric mirror, an optical position detector, and a stator driven by an analog PID servo amplifier. The optical path is directed first onto the X-axis mirror, deflected at a right angle toward the orthogonal Y-axis mirror, and projected outward into atmospheric space.
The angular displacement \theta(t) of a galvanometer mirror is governed by a second-order torsional differential equation:
J \frac{d^2\theta(t)}{dt^2} + B \frac{d\theta(t)}{dt} + K \theta(t) = \tau(t) = K_t \, i(t)
where J is the combined rotational moment of inertia of the magnet rotor, shaft, and glass mirror substrate; B is the mechanical and back-EMF damping coefficient; K is the torsional restoring spring constant; K_t is the electromechanical motor torque constant; and i(t) is the instantaneous drive current supplied by the servo amplifier. In standard state-variable form, normalizing by inertia yields:
\ddot{\theta}(t) + 2\zeta \omega_n \dot{\theta}(t) + \omega_n^2 \theta(t) = \omega_n^2 u(t)
where \omega_n = \sqrt{K/J} is the undamped angular natural frequency and \zeta = B / (2\sqrt{J K}) is the dimensionless damping ratio.
Industrial laser projectors are calibrated according to ILDA specifications, typically rated at 30\text{ kpps} (kilo-points per second) or 40\text{ kpps} at an 8^\circ optical scan angle. When the control software dispatches an instantaneous position step command u(t) = \Theta, the physical scanner mirror exhibits finite rise time and settling behavior. The system step response partitions into three operating regimes depending on servo amplifier tuning:
In an underdamped system (\zeta < 1), the mirror accelerates rapidly toward the target setpoint but overshoots, exhibiting damped harmonic ringing before stabilizing:
\theta(t) = \Theta \left[1 - \frac{e^{-\zeta \omega_n t}}{\sqrt{1-\zeta^2}} \sin\left(\omega_d t + \phi\right) \right]
where \omega_d = \omega_n \sqrt{1 - \zeta^2} and \phi = \arccos(\zeta). The peak percentage overshoot is expressed as:
$$M_p = \exp\left(-\frac{\pi \zeta}{\sqrt{1 - \zeta^2}} \right)[span_68](start_span)[span_68](end_span)
In visual displays, an underdamped scanner causes sharp vector corners to visibly hook, vibrate, and deform. Conversely, in an overdamped system (\zeta > 1), the servo suppresses all oscillation, but mirror inertia introduces severe phase lag, rounding sharp geometric corners into sluggish parabolic curves. A critically damped system (\zeta = 1) achieves the fastest transition

to the commanded setpoint without overshoot:

$$\theta(t) = \Theta \left[1 - e^{-\omega_n t} (1 + \omega_n t) \right]$$

Visual Manifestations of Galvo Dynamics

Physical laser projectors display distinct visual artifacts resulting directly from scanner inertia and laser diode modulation timing. A prominent artifact is corner dwell light accumulation. Because mirrors must decelerate to zero angular velocity to reverse direction at sharp vector vertices, the beam lingers longer at vertices than along connecting line segments. Integrated optical exposure is inversely proportional to mirror scanning velocity:

$$\text{Expo[span_71](start_span)[span_71](end_span)sure}(x, y) \propto \frac{1}{\Vert{}\vec{v}(t)\Vert{}}$$

This velocity drop creates brilliant, pin-point nodes of light at geometric vertices, known in laser display physics as corner dwell. Additionally, modulating laser output synchronously with high-speed mirror movement requires microsecond-level synchronization. Latency mismatches between diode drivers and mechanical deflection produce visible tails or hooks at vector termination points.

Kinematic Projection Modes

To replicate an authentic concert rig, the simulation engine computes beam endpoints using physical scanner transformations rather than static 2D vector drift.

In sweeping fan mode, an array of discrete beams generated by time-division multiplexing or multi-emitter arrays is swept through angular sinusoidal fields:

$$\theta_i(t) = A_x \sin\left(\omega_{sweep} t + \phi_i\right) + \text{noise}(t)$$

where individual beams maintain phase relationships ($\phi_i = i \cdot \frac{2\pi}{N}$) to create oscillating wave fans that slice through theatrical haze.

In conic liquid-sky mode, a beam is scanned horizontally at high frequencies ($f_x \ge 500\text{ Hz}$), exceeding the human critical flicker fusion threshold ($\sim 60\text{ Hz}$) to form an apparently solid planar sheet of light. When pitched slightly downward across the vehicle chassis, the plane cuts horizontally through rolling fog. Modulating the vertical mirror with a low-frequency sinusoidal offset:

$$x(\alpha) = R \cos(\alpha), \quad y(\alpha) = R \sin(\alpha) \cdot \sin(\psi) + Y_{tilt}, \quad \alpha \in [0, 2\pi)$$

causes the planar sheet to deform into a sweeping dynamic cone, creating the optical illusion of a floating, undulating liquid ceiling.

In ballistic snap-targeting mode, the scanners transition between display cues by executing rapid jumps at maximum slewing acceleration. The motion is characterized by an initial ballistic impulse followed by a critically damped deceleration phase, causing beams to flick instantaneously between stage focal points, snap to fixed target angles, and hold steady with microscopic servo jitter.

Procedural Chassis Smoke and Dynamic Volumetric Flare Interaction

Procedural Turbulence Generation

Simulating chassis-mounted smoke or stage haze requires an efficient, real-time procedural density field \rho(x, y, t). While solving full 2D Navier-Stokes equations is computationally prohibitive in a real-time web rendering loop, a continuous, drifting turbulence field can be generated using multi-octave Fractal Brownian Motion (fBm) layered over a particulate physics buffer. The scalar smoke density \rho at any coordinate is defined by summing O octaves of continuous gradient or value noise:

$$\rho(x, y, t) = \sum_{k=0}^{O-1} \gamma^k \, \mathcal{N}\left(2^k \cdot \frac{x - v_x t}{\lambda}, \; 2^k \cdot \frac{y - v_y t}{\lambda} \right)$$

where \gamma \approx 0.5 is the persistence factor, \lambda is the fundamental spatial wavelength, \vec{v} = (v_x, v_y) represents the ambient advection velocity (wind draft along the vehicle chassis), and \mathcal{N} is a continuous 2D procedural noise function.

To enhance realism, this continuous density field is paired with an array of drifting macroscopic smoke cluster particulates that undergo turbulent advection:

$$\vec{x}_p(t + \Delta t) = \vec{x}_p(t) + \left(\vec{v}_{wind} + \vec{v}_{turb}(\vec{x}_p) \right) \Delta t$$

$$\vec{v}_{turb}(x, y) = \left(\frac{\partial \psi}{\partial y}, \, -\frac{\partial \psi}{\partial x} \right)$$

where \psi(x, y) is a scalar stream function derived from the underlying noise lattice, guaranteeing divergence-free, incompressible turbulent eddy advection.

Volumetric Ray-Marching and Beam-Smoke Coupling

As a laser beam traverses the smoke field from its origin \vec{P}_{origin} to its scanner target \vec{P}_{target}, scattered intensity varies dynamically in proportion to local aerosol density. The total scattered power S along an incremental path segment ds is given by:

$$d S(s) = I(s) \cdot \rho(\vec{r}(s)) \cdot \Phi(\theta) \, ds$$

where \vec{r}(s) = \vec{P}_{origin} + s \cdot \hat{u} is the position vector along the beam direction \hat{u}, \rho(\vec{r}(s)) is the procedural smoke density sampled at that location, and \Phi(\theta) is the forward scattering phase constant.

In the canvas rendering pipeline, this integral is approximated using discrete stochastic line sampling. The engine marches along each active laser vector, evaluating equidistant sample points against the smoke density field. Where local aerosol density is low, the beam propagates cleanly with standard triple-pass attenuation. However, when the sampled density exceeds an activation threshold, the local parcel is identified as a dense particulate pocket. At these locations, the engine computes a dynamic flare coefficient:

$$\eta_k = \text{clamp}\left(\frac{\rho(\vec{r}_k) - \rho_{thresh}}{1 - \rho_{thresh}}, \, 0, \, 1 \right)$$

This coefficient scales a localized radial flare rasterized with an additive two-stop radial gradient: the center radiates the pure white hot-core value, smoothly decaying into the saturated spectral beam color, and fading to zero alpha at outer radius $R_{flare} \propto \eta_k$. Concurrently, the width and opacity of the laser beam's mid-spectral pass widen through the pocket, visually mimicking intense forward scattering through thick aerosol billows.

Production Implementation: Unified Laser Engine and Audio Handler

The following implementation unifies the non-preemptive audio pipeline, second-order

galvanometer kinematics solver, procedural chassis smoke field generator, and multi-pass additive laser rasterizer into a high-performance system.

```
/**
 * Concert-Grade Subaru-Spec Laser Rig & Non-Preemptive Audio Engine
 * Architecture:
 *   1. Non-Preemptive Web Audio Pipeline (Bypasses Android Exclusive
Focus)
 *   2. Second-Order Torsional ILDA Galvanometer Kinematics Solver
 *   3. Multi-Octave Chassis Smoke / Aerosol Density Field
 *   4. Triple-Pass Additive Photometric Laser Rasterizer
 */

// MODULE 1: NON-PREEMPTIVE AUDIO PIPELINE HANDLER

class NonPreemptiveAudioHandler {
  constructor(audioSrc) {
    this.audioSrc = audioSrc;
    this.audioElement = null;
    this.audioCtx = null;
    this.isInitialized = false;
    this.isPlaying = false;
  }

  async initializeAndPlay() {
    if (this.isPlaying) return;

    try {
      if ('audioSession' in navigator && navigator.audioSession) {
        navigator.audioSession.type = 'ambient';
      }

      if (!this.audioElement) {
        this.audioElement = document.createElement('audio');
        this.audioElement.preload = 'none';
        this.audioElement.src = this.audioSrc;
        this.audioElement.loop = true;
        this.audioElement.playsInline = true;

        this.audioElement.disableRemotePlayback = true;
        this.audioElement.setAttribute('disableremoteplayback',
'true');
        this.audioElement.setAttribute('webkit-playsinline', 'true');

        this.audioElement.style.display = 'none';
        document.body.appendChild(this.audioElement);
      }

      if ('mediaSession' in navigator) {
```

```
        navigator.mediaSession.playbackState = 'none';
        const noopActions = ['play', 'pause', 'seekbackward',
'seekforward', 'previoustrack', 'nexttrack'];
        noopActions.forEach(action => {
          try { navigator.mediaSession.setActionHandler(action, null);
} catch (e) {}
        });
      }

      const AudioContextClass = window.AudioContext ||
window.webkitAudioContext;
      if (AudioContextClass) {
        if (!this.audioCtx) {
          this.audioCtx = new AudioContextClass();
        }
        if (this.audioCtx.state === 'suspended') {
          await this.audioCtx.resume();
        }
      }

      await this.audioElement.play();
      this.isPlaying = true;
      this.isInitialized = true;

      if ('mediaSession' in navigator) {
        navigator.mediaSession.playbackState = 'none';
      }
    } catch (err) {
      console.warn('[AudioHandler] Non-preemptive initialization
failed:', err);
    }
  }

  stop() {
    if (this.audioElement && this.isPlaying) {
      this.audioElement.pause();
      this.isPlaying = false;
    }
    if (this.audioCtx && this.audioCtx.state === 'running') {
      this.audioCtx.suspend();
    }
    if ('mediaSession' in navigator) {
      navigator.mediaSession.playbackState = 'none';
    }
  }
}

// MODULE 2: SECOND-ORDER GALVANOMETER KINEMATICS ENGINE
```

```
class GalvoScanner {
  constructor(options = {}) {
    this.x = options.initialX || 0;
    this.y = options.initialY || 0;
    this.vx = 0;
    this.vy = 0;
    this.targetX = this.x;
    this.targetY = this.y;

    this.omega_n = options.omega_n || 65.0;
    this.zeta = options.zeta || 0.82;
    this.speed = 0;
  }

  setTarget(tx, ty) {
    this.targetX = tx;
    this.targetY = ty;
  }

  update(dt) {
    const h = Math.min(dt, 0.033);

    const ax = -2.0 * this.zeta * this.omega_n * this.vx -
(this.omega_n ** 2) * (this.x - this.targetX);
    this.vx += ax * h;
    this.x += this.vx * h;

    const ay = -2.0 * this.zeta * this.omega_n * this.vy -
(this.omega_n ** 2) * (this.y - this.targetY);
    this.vy += ay * h;
    this.y += this.vy * h;

    this.speed = Math.sqrt(this.vx * this.vx + this.vy * this.vy);
  }
}

// MODULE 3: PROCEDURAL CHASSIS SMOKE & TURBULENCE SIMULATOR

class ChassisSmokeField {
  constructor(width, height) {
    this.width = width;
    this.height = height;
    this.particles = [];
    this.maxParticles = 65;
    this.time = 0;
    this.initParticles();
  }
```

```
initParticles() {
  for (let i = 0; i < this.maxParticles; i++) {
    this.particles.push({
      x: Math.random() * this.width,
      y: this.height * 0.4 + Math.random() * (this.height * 0.6),
      vx: 15.0 + Math.random() * 25.0,
      vy: (Math.random() - 0.5) * 8.0,
      radius: 35.0 + Math.random() * 55.0,
      density: 0.15 + Math.random() * 0.35,
      phase: Math.random() * Math.PI * 2
    });
  }
}

resize(width, height) {
  this.width = width;
  this.height = height;
}

update(dt) {
  this.time += dt;
  for (let i = 0; i < this.particles.length; i++) {
    const p = this.particles[i];
    p.x += (p.vx + Math.sin(this.time * 1.5 + p.phase) * 12.0) * dt;
    p.y += (p.vy + Math.cos(this.time * 1.2 + p.phase) * 6.0) * dt;

    if (p.x - p.radius > this.width) {
      p.x = -p.radius;
      p.y = this.height * 0.4 + Math.random() * (this.height * 0.6);
    }
  }
}

sampleDensity(px, py) {
  let accumulatedDensity = 0.0;
  for (let i = 0; i < this.particles.length; i++) {
    const p = this.particles[i];
    const dx = px - p.x;
    const dy = py - p.y;
    const distSq = dx * dx + dy * dy;
    const rSq = p.radius * p.radius;
    if (distSq < rSq) {
      const falloff = 1.0 - distSq / rSq;
      accumulatedDensity += p.density * (falloff ** 2);
    }
  }
  return Math.min(accumulatedDensity, 1.0);
```

```
  }

  draw(ctx) {
    ctx.save();
    ctx.globalCompositeOperation = 'screen';
    for (let i = 0; i < this.particles.length; i++) {
      const p = this.particles[i];
      const grad = ctx.createRadialGradient(p.x, p.y, 0, p.x, p.y,
p.radius);
      grad.addColorStop(0, `rgba(160, 185, 200, ${p.density *
0.25})`);
      grad.addColorStop(0.6, `rgba(120, 140, 160, ${p.density *
0.12})`);
      grad.addColorStop(1, 'rgba(0, 0, 0, 0)');

      ctx.fillStyle = grad;
      ctx.beginPath();
      ctx.arc(p.x, p.y, p.radius, 0, Math.PI * 2);
      ctx.fill();
    }
    ctx.restore();
  }
}

// MODULE 4: CONCERT-GRADE LASER RIG RASTERIZER

class LaserRigRenderer {
  constructor(canvasElement) {
    this.canvas = canvasElement;
    this.ctx = canvasElement.getContext('2d');
    this.smoke = new ChassisSmokeField(canvasElement.width,
canvasElement.height);

    this.greenGalvo = new GalvoScanner({ omega_n: 68.0, zeta: 0.85 });
    this.redGalvo = new GalvoScanner({ omega_n: 62.0, zeta: 0.78 });

    this.mountOrigins = {
      green: { x: this.canvas.width * 0.35, y: this.canvas.height *
0.45 },
      red:   { x: this.canvas.width * 0.65, y: this.canvas.height *
0.45 }
    };

    this.elapsedTime = 0;
    this.lastTimestamp = performance.now();
  }

  resize(w, h) {
```

```
  this.canvas.width = w;
  this.canvas.height = h;
  this.smoke.resize(w, h);
  this.mountOrigins.green = { x: w * 0.35, y: h * 0.45 };
  this.mountOrigins.red = { x: w * 0.65, y: h * 0.45 };
}

renderTriplePassBeam(x1, y1, x2, y2, colorSpec) {
  const ctx = this.ctx;

  // Pass 1: Outer Volumetric Haze
  ctx.save();
  ctx.strokeStyle = colorSpec.haze;
  ctx.lineWidth = 20.0;
  ctx.lineCap = 'round';
  ctx.shadowColor = colorSpec.shadow;
  ctx.shadowBlur = 18.0;
  ctx.beginPath();
  ctx.moveTo(x1, y1);
  ctx.lineTo(x2, y2);
  ctx.stroke();
  ctx.restore();

  // Pass 2: Mid Spectral Beam
  ctx.save();
  ctx.strokeStyle = colorSpec.spectral;
  ctx.lineWidth = 6.0;
  ctx.lineCap = 'round';
  ctx.beginPath();
  ctx.moveTo(x1, y1);
  ctx.lineTo(x2, y2);
  ctx.stroke();
  ctx.restore();

  // Pass 3: Piercing Hot Core
  ctx.save();
  ctx.strokeStyle = '#ffffff';
  ctx.lineWidth = 1.5;
  ctx.lineCap = 'round';
  ctx.beginPath();
  ctx.moveTo(x1, y1);
  ctx.lineTo(x2, y2);
  ctx.stroke();
  ctx.restore();
}

renderBeamSmokeInteraction(x1, y1, x2, y2, colorSpec) {
  const steps = 14;
```

```
    const dx = (x2 - x1) / steps;
    const dy = (y2 - y1) / steps;

    for (let i = 1; i < steps; i++) {
      const sampleX = x1 + dx * i;
      const sampleY = y1 + dy * i;
      const density = this.smoke.sampleDensity(sampleX, sampleY);

      if (density > 0.22) {
        const flareIntensity = (density - 0.22) / 0.78;
        const flareRadius = 8.0 + flareIntensity * 28.0;

        const grad = this.ctx.createRadialGradient(
          sampleX, sampleY, 0,
          sampleX, sampleY, flareRadius
        );
        grad.addColorStop(0, '#ffffff');
        grad.addColorStop(0.3, colorSpec.spectral);
        grad.addColorStop(0.8, colorSpec.haze);
        grad.addColorStop(1, 'rgba(0, 0, 0, 0)');

        this.ctx.fillStyle = grad;
        this.ctx.beginPath();
        this.ctx.arc(sampleX, sampleY, flareRadius, 0, Math.PI * 2);
        this.ctx.fill();
      }
    }
  }

  renderCornerDwellPoint(x, y, speed, colorSpec) {
    const dwellFactor = Math.max(0.2, 1.0 - Math.min(speed / 1800.0,
0.8));
    const radius = 2.5 * dwellFactor;

    this.ctx.save();
    this.ctx.fillStyle = '#ffffff';
    this.ctx.shadowColor = colorSpec.spectral;
    this.ctx.shadowBlur = 12.0 * dwellFactor;
    this.ctx.beginPath();
    this.ctx.arc(x, y, radius, 0, Math.PI * 2);
    this.ctx.fill();
    this.ctx.restore();
  }

  updateGalvoTargets(t) {
    const w = this.canvas.width;
    const h = this.canvas.height;
    const modeCycle = (t * 0.2) % 3.0;
```

```
    if (modeCycle < 1.0) {
      const sweepAngle = Math.sin(t * 3.2) * (w * 0.38);
      this.greenGalvo.setTarget(w * 0.5 + sweepAngle, h * 0.88);
      this.redGalvo.setTarget(w * 0.5 - sweepAngle * 0.85, h * 0.92);
    } else if (modeCycle < 2.0) {
      const liquidSkyPhase = t * 14.0;
      const coneRadius = w * 0.42;
      const elevationTilt = Math.sin(t * 1.5) * (h * 0.08);

      const gx = w * 0.5 + Math.cos(liquidSkyPhase) * coneRadius;
      const gy = h * 0.72 + Math.sin(liquidSkyPhase) * (h * 0.04) +
elevationTilt;

      const rx = w * 0.5 - Math.cos(liquidSkyPhase * 0.95) *
coneRadius;
      const ry = h * 0.74 - Math.sin(liquidSkyPhase * 0.95) * (h *
0.04) + elevationTilt;

      this.greenGalvo.setTarget(gx, gy);
      this.redGalvo.setTarget(rx, ry);
    } else {
      const snapIndex = Math.floor(t * 1.8) % 4;
      const targets = [
        { gx: w * 0.15, gy: h * 0.85, rx: w * 0.85, ry: h * 0.85 },
        { gx: w * 0.80, gy: h * 0.75, rx: w * 0.20, ry: h * 0.75 },
        { gx: w * 0.30, gy: h * 0.95, rx: w * 0.70, ry: h * 0.95 },
        { gx: w * 0.50, gy: h * 0.65, rx: w * 0.50, ry: h * 0.65 }
      ];
      const jitterX = (Math.random() - 0.5) * 3.0;
      const jitterY = (Math.random() - 0.5) * 3.0;
      this.greenGalvo.setTarget(targets[snapIndex].gx + jitterX,
targets[snapIndex].gy + jitterY);
      this.redGalvo.setTarget(targets[snapIndex].rx - jitterX,
targets[snapIndex].ry - jitterY);
    }
  }

  renderFrame() {
    const currentTimestamp = performance.now();
    const dt = (currentTimestamp - this.lastTimestamp) / 1000.0;
    this.lastTimestamp = currentTimestamp;
    this.elapsedTime += dt;

    const ctx = this.ctx;
    const w = this.canvas.width;
    const h = this.canvas.height;
```

```
    ctx.save();
    ctx.globalCompositeOperation = 'source-over';
    ctx.fillStyle = 'rgba(5, 7, 10, 0.4)';
    ctx.fillRect(0, 0, w, h);
    ctx.restore();

    this.smoke.update(dt);
    this.smoke.draw(ctx);

    this.updateGalvoTargets(this.elapsedTime);
    this.greenGalvo.update(dt);
    this.redGalvo.update(dt);

    const greenSpec = {
      haze: 'rgba(0, 255, 68, 0.18)',
      spectral: '#00ff44',
      shadow: '#00ff44'
    };
    const redSpec = {
      haze: 'rgba(255, 17, 51, 0.18)',
      spectral: '#ff1133',
      shadow: '#ff1133'
    };

    ctx.save();
    ctx.globalCompositeOperation = 'lighter';

    const gx1 = this.mountOrigins.green.x;
    const gy1 = this.mountOrigins.green.y;
    const gx2 = this.greenGalvo.x;
    const gy2 = this.greenGalvo.y;
    this.renderTriplePassBeam(gx1, gy1, gx2, gy2, greenSpec);
    this.renderBeamSmokeInteraction(gx1, gy1, gx2, gy2, greenSpec);
    this.renderCornerDwellPoint(gx2, gy2, this.greenGalvo.speed,
greenSpec);

    const rx1 = this.mountOrigins.red.x;
    const ry1 = this.mountOrigins.red.y;
    const rx2 = this.redGalvo.x;
    const ry2 = this.redGalvo.y;
    this.renderTriplePassBeam(rx1, ry1, rx2, ry2, redSpec);
    this.renderBeamSmokeInteraction(rx1, ry1, rx2, ry2, redSpec);
    this.renderCornerDwellPoint(rx2, ry2, this.redGalvo.speed,
redSpec);

    const auxCount = 5;
    for (let i = 0; i < auxCount; i++) {
      const offset = (i - (auxCount - 1) / 2) * (w * 0.08);
```

```
      const auxTargetX = this.greenGalvo.x + offset;
      const auxTargetY = this.greenGalvo.y + Math.abs(offset) * 0.15;

      const bladeSpec = {
        haze: 'rgba(0, 255, 68, 0.08)',
        spectral: 'rgba(0, 255, 68, 0.45)',
        shadow: '#00ff44'
      };
      this.renderTriplePassBeam(gx1, gy1, auxTargetX, auxTargetY,
bladeSpec);
    }

    ctx.restore();

    requestAnimationFrame(() => this.renderFrame());
  }
}

// SYSTEM BOOTSTRAPPER AND EVENT BINDING

window.addEventListener('DOMContentLoaded', () => {
  const canvas = document.getElementById('laserCanvas') || (() => {
    const c = document.createElement('canvas');
    c.id = 'laserCanvas';
    c.style.position = 'fixed';
    c.style.top = '0';
    c.style.left = '0';
    c.style.width = '100vw';
    c.style.height = '100vh';
    c.style.zIndex = '0';
    c.style.background = '#020305';
    document.body.appendChild(c);
    return c;
  })();

  canvas.width = window.innerWidth;
  canvas.height = window.innerHeight;

  const audioEngine = new
NonPreemptiveAudioHandler('audio/intro_track.mp3');
  const laserEngine = new LaserRigRenderer(canvas);

  window.addEventListener('resize', () => {
    laserEngine.resize(window.innerWidth, window.innerHeight);
  });

  const activateBtn = document.getElementById('introBtn') || (() => {
    const btn = document.createElement('button');
```

```
    btn.id = 'introBtn';
    btn.innerText = 'Engage Subaru-Spec Rig';
    btn.style.position = 'fixed';
    btn.style.bottom = '30px';
    btn.style.left = '50%';
    btn.style.transform = 'translateX(-50%)';
    btn.style.padding = '14px 28px';
    btn.style.backgroundColor = '#00ff44';
    btn.style.color = '#000000';
    btn.style.fontFamily = 'monospace';
    btn.style.fontWeight = 'bold';
    btn.style.fontSize = '14px';
    btn.style.border = 'none';
    btn.style.borderRadius = '4px';
    btn.style.cursor = 'pointer';
    btn.style.zIndex = '100';
    btn.style.boxShadow = '0 0 20px rgba(0, 255, 68, 0.6)';
    document.body.appendChild(btn);
    return btn;
  })();

  activateBtn.addEventListener('click', async () => {
    await audioEngine.initializeAndPlay();
    activateBtn.style.display = 'none';
  });

  laserEngine.renderFrame();
});
```

Conclusions and Systems Integration Matrix

Deploying a stage-grade visual laser simulation within standard web runtimes requires strict decoupling between client-side graphics rendering and mobile operating system audio arbitration. The interaction between continuous hardware-accelerated canvas contexts and underlying platform media subsystems produces specific engineering trade-offs that dictate deployment architecture:

Subsystem Integration Matrix

Subsystem Component	Key Technical Vulnerability	Implemented Resolution Architecture	Architectural Verification Criteria
Android Audio Concurrency	Blink grabs AUDIOFOCUS_GAIN on unmuted audio tags, pausing YouTube Music.	preload="none", disableRemotePlayback=true, and navigator.audioSession.type=[span_92](start_s	Background music continues uninterrupted during page visit and explicit activation.

Subsystem Component	Key Technical Vulnerability	Implemented Resolution Architecture	Architectural Verification Criteria
		pan)[span_92](end_span)[span_93](start_span)[span_93](end_span)[span_94](start_span)[span_94](end_span)"ambient".	
Blink Lock-Screen Sync	Chromium registers media controls with the OS Universal Media Object notification.	Force navigator.mediaSession.playbackState = 'none' and nullify action handlers.	Lock screen displays background music player album art rather than browser transport.
Photon Vector Radiance	Standard alpha compositing yields muddy, opaque overlapping line intersections.	Canvas pipeline configured to globalCompositeOperation = 'lighter' (additive mode).	Intersecting green and red vectors generate pure white/yellow chromatic blooming nodes.
Atmospheric Realism	Single-stroke 2D vector primitives appear flat, artificial, and devoid of depth.	Triple-pass geometry: 20px volumetric haze, 6px spectral line, and 1.5px achromatic core.	Beams exhibit physical Mie forward scattering with a concentrated optical core.
Scanner Kinematics	Linear coordinate interpolation appears synthetic, lacking mechanical inertia.	Second-order torsional differential equations with step response overshoot and corner dwell.	Mirrors show realistic physical lag, damping overshoot, and velocity-dependent corner light studs.
Chassis Smoke Coupling	Static line overlays ignore theatrical fog density variations.	Line-integral stochastic sampling over procedural advecting smoke particulate fields.	Beams flare with radial explosive blooms precisely when cutting through smoke pockets.

By executing laser rendering exclusively via three-tier additive vector passes over a procedural particulate field, the canvas accurately models the real-world optics of high-flux coherent beams traversing forward-scattering aerosol suspensions. Concurrently, strictly enforcing preload="none", disabling remote playback UI, tearing down MediaSession lock-screen registrations, and leveraging ambient audio types eliminates the invasive audio focus preemption that traditionally compromises web-based multimedia experiences on Android devices. This unified architecture guarantees concert-grade visual fidelity while preserving full system-level audio coexistence.

Works cited

1. Manage audio focus | Android media, https://developer.android.com/media/optimize/audio-focus 2. Android : How to stop music service of my app, if another app plays, https://stackoverflow.com/questions/24716455/android-how-to-stop-music-service-of-my-app-if-

another-app-plays-music 3. A WebView that streams live video requests audio focus while in the, https://issues.chromium.org/40946745 4. Background audio hardening - Android Developers, https://developer.android.com/about/versions/17/changes/bg-audio 5. Customize media notifications and playback controls with ... - web.dev, https://web.dev/articles/media-session 6. Google Cast for Chrome on Android | Blog, https://developer.chrome.com/blog/presentation-api 7. HTMLMediaElement - Web APIs | MDN, https://developer.mozilla.org/en-US/docs/Web/API/HTMLMediaElement 8. AudioSession - Web APIs | MDN, https://developer.mozilla.org/en-US/docs/Web/API/AudioSession 9. media-chrome/CHANGELOG.md at main - muxinc - GitHub, https://github.com/muxinc/media-chrome/blob/main/CHANGELOG.md 10. Android Chrome does not allow applications to play HTML5 audio, https://issues.chromium.org/40303049 11. Remote Playback API - MDN Web Docs, https://developer.mozilla.org/en-US/docs/Web/API/Remote_Playback_API 12. Audio Session API - MDN Web Docs - Mozilla, https://developer.mozilla.org/en-US/docs/Web/API/Audio_Session_API 13. AudioSession: type property - Web APIs | MDN, https://developer.mozilla.org/en-US/docs/Web/API/AudioSession/type 14. audio-session/explainer.md at main - GitHub, https://github.com/w3c/audio-session/blob/main/explainer.md 15. 25 Kpps laser scanning galvo galvanometer scanners Set ILDA, https://beamq.com/laser/25-kpps-laser-scanning-galvo-galvanometer-scanners-set-ilda/ 16. Laser Scanning Galvanometer: Principles, Control, and Advanced, https://www.ezcad.com/laser-scanning-galvanometer-from-core-principles-to-advanced-control/ 17. stoatworks-labs/galvo: An ILDA laser projector as an FFGL effect for, https://github.com/stoatworks-labs/galvo 18. ILDA Test Pattern for Laser Show Lights, https://www.starshinelights.com/blogs/news/ilda-test-pattern-laser-show-lights 19. Laser Scanners and the ILDA Test Pattern, https://nfllaser.com/blogs/all-articles/laser-scanners-and-the-ilda-test-pattern 20. Measurements of galvanometer step response. (a) Experimental, https://www.researchgate.net/figure/Measurements-of-galvanometer-step-response-a-Experimental-setup-A-DAQ-device_fig1_354982088 21. Question about Galvo Speed? | Laser Pointer Forums, https://laserpointerforums.com/threads/question-about-galvo-speed.71998/