SAPIENZA UNIVERSITY OF ROME A.Y. 2025/2026

FOURIER CITY

An Interactive, Audio-Reactive City

Technical report and user manual for the Interactive Graphics course

Abstract. Fourier City creates a three-dimensional graphical representation of sound in its two main forms: the time-domain waveform and the frequency-domain response. Users interact with sound in real time and observe how pitch and filtering alter both what they hear and what they see. The result is an explorable city where buildings, lines, controls, light and a robotic arm turn signal-processing concepts into visible behavior.

Project Intentions

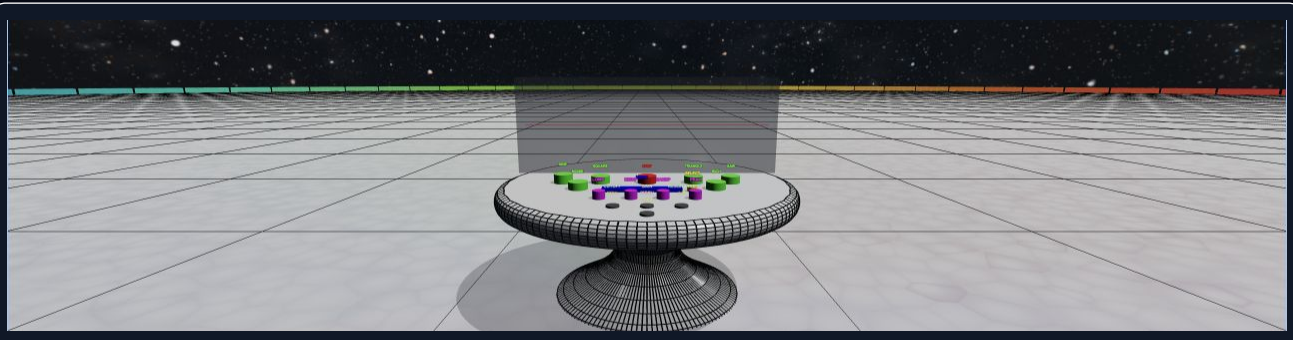
The environment is designed as an interactive interpretation of sound. Periodic signals, frequency components, filters and audio transformations are represented through buildings, animated lines, buttons, knobs, a waveform display and a hierarchical robot. Each element has an explanatory role: the scene responds to the same signal that reaches the listener.

The main intention is to make Fourier analysis, frequency response, filtering and periodic waveforms more accessible. Instead of presenting these ideas only as equations or static plots, Fourier City lets the user hear a change, see its effect and move around the visualization.

TIME DOMAIN
The live waveform shows how amplitude evolves over time.

FREQUENCY DOMAIN
The skyline reveals how energy is distributed across the spectrum.

INTERACTION
Physical controls connect audible changes with immediate visual feedback.



THE CITY AS A SPECTRUM — time, frequency and interaction in one scene

AUTHORS
Lorenzo Marinelli
Alessandro Romania

COURSE
Interactive Graphics
Prof. Marco Schaerf
DIAG, Sapienza University of Rome

User Manual: Starting and Exploring

This manual describes what the scene represents and how to use it. Technical details are introduced later; the purpose here is to help a first-time user understand the experience.

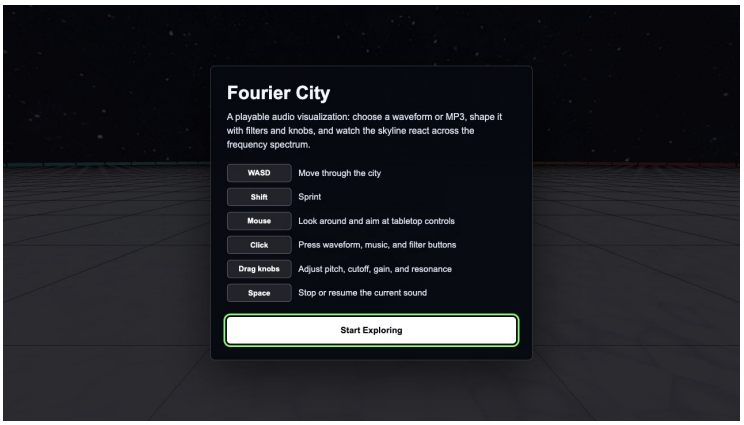


Figure 1: The initial screen summarizes the essential controls.

Starting the Scene

1. Run the project and open the address printed by Vite.
2. Read the initial panel and press **Start Exploring**. This gesture enables browser audio and requests mouse capture.
3. The default music source begins automatically. Aim at the table to select another sound or filter.
4. Press Escape to release the mouse. Click the scene to resume exploration.

The **SELECT** button opens a local MP3 chooser. Loading a file is optional; every procedural signal works without one.

Moving Through the Environment

The camera uses a first-person viewpoint. The table and robot are solid obstacles; blocked diagonal movement slides along their edge rather than passing through them.

W / A / S / D	Walk in four directions.	**Mouse**	Look and aim the crosshair.
Left Shift	Sprint while moving.	**Click**	Press a button or drag a knob.
Vertical drag	Change the aimed knob value.	**Space**	Pause or resume sound.
Escape	Release pointer lock or close the intro.	**Scene click**	Capture the mouse again.

What the Main Objects Represent

FREQUENCY SKYLINE

Buildings are logarithmic frequency bands; height and brightness follow energy. The **red curve** predicts the active filter and its **yellow marker** identifies cutoff.

CONTROL DESK

Buttons choose sources and filters, open MP3 selection, or pause playback. **Knobs** change pitch, cutoff, gain and resonance with numeric feedback.

TIME AND FLOW

The **waveform glass** shows filtered amplitude over time. The animated **ground flow** carries spectrum color and energy outward from the source.

PITCH ROBOT

For a periodic signal, the **robotic arm** points to the skyline band that matches the selected pitch and follows that building's roof.

How to read the city. Position and hue identify frequency from low (blue/cyan) to high (red/magenta); height, light and motion identify energy. Filtering changes both sound and visualization because analysis occurs after the filter.

User Manual: Sounds, Filters and Controls

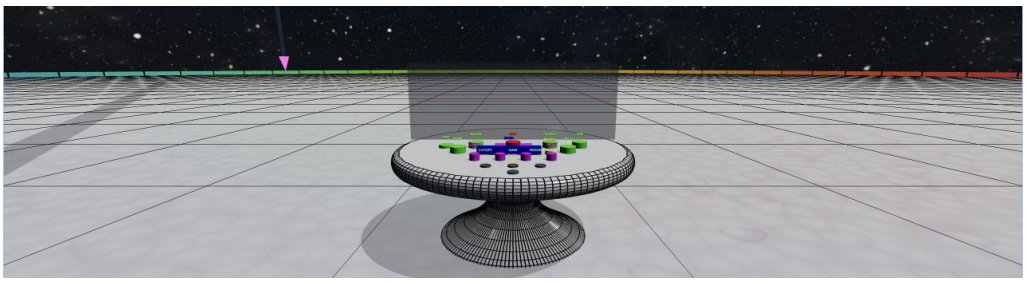


Figure 2: The in-world table groups source, filter and parameter controls.

Sounds and Main Functions

Sine	Pure fundamental frequency.
Square	Bright odd-harmonic spectrum.
Triangle	Softer, rapidly decaying odd harmonics.
Saw	Dense spectrum of integer harmonics.
Rich	Eight weighted harmonics.
Noise	Broadband signal without one pitch.
Music	Bundled or locally selected MP3.

Selecting a source replaces the previous one, resets pause and updates pitch. **STOP/RESUME** and Space share the same state; active buttons remain depressed.

Knob Ranges

Pitch	MP3: 0.50 to 1.50x; periodic: 20 to 20,000 Hz; noise: N/A.
Cutoff	20 to 20,000 Hz, logarithmic.
Gain	-18 to +18 dB for peaking.
Resonance	$Q = 0.1$ to 20; controls emphasis and width.

Vertical drag rotates a knob through 270 degrees, redraws its label and updates sound and visualization. Periodic pitch also moves the robot target.

Available Filters

Filter	Audible effect	Visual response
Low-pass	Keeps low frequencies and progressively reduces frequencies above cutoff.	Red curve stays high on the low-frequency side and falls toward the right.
High-pass	Reduces low frequencies while allowing frequencies above cutoff to pass.	Red curve rises after the yellow cutoff marker.
Band-pass	Keeps a region around cutoff and reduces frequencies outside it.	Red curve forms a peak whose width depends on resonance.
Peaking	Boosts or cuts a region around cutoff without removing the rest of the spectrum.	Red curve forms a positive or negative bell controlled by gain and resonance.

Press the selected filter again to bypass it. Cutoff and resonance affect all four modes; gain is meaningful for peaking. There is no band-stop mode in the current version.

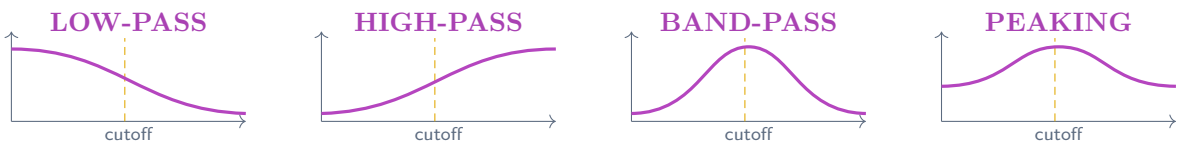


Figure 3: Simplified filter shapes. The yellow dashed line represents cutoff.

Technical Report: Libraries, Camera and Scene

Libraries and Tools

Technology	Role in Fourier City
Three.js 0.184	Scene graph, WebGL renderer, geometry, materials, math, positional audio and animation infrastructure [1].
Three.js add-ons	`PointerLockControls`, `OBJLoader`, `EXRLoader`, wide-line classes and `BufferGeometryUtils`.
Web Audio API	Audio buffers, oscillators, periodic waves, biquad filters, time-domain samples and FFT analysis [2].
Vite 8 / Node.js / npm	Development server, native ES-module resolution, production bundling and project scripts [3].
Blender 5.0	Creation and OBJ export of the original table and skyscraper models [4].
Git / GitHub	Version control, collaboration and project publication.

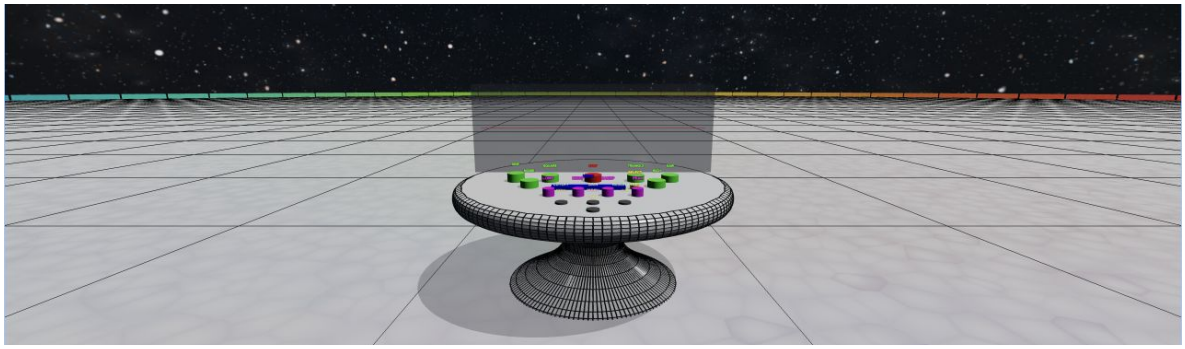


Figure 4: The table, waveform display and frequency-colored skyline establish the scene's visual axis.

Camera and Renderer

The first-person camera uses a 75° field of view, 0.1/1000 near and far planes, and a fixed 2.3-unit eye height. `PointerLockControls` converts mouse motion into view direction; resize events update renderer size, pixel ratio, aspect and projection. The renderer enables antialiasing, sRGB output, physically correct lights, ACES filmic tone mapping and shadow maps. Movement follows the horizontal camera basis at 3.8 units/s or 7 while sprinting. Analytic table and robot collisions allow axis-by-axis sliding.

Models and Structure

The table and skyscraper are original Blender models exported as OBJ geometry. The table loads as solid and wireframe layers. Skyline geometry is normalized, merged and reused through instancing.
Buttons, knobs, floor, waveform glass and robot parts use Three.js primitives. This mix keeps authored silhouettes where they matter while allowing interactive objects and hierarchies to be generated directly in JavaScript.

Materials and Lights

Standard, physical, emissive, transparent and shader materials distinguish surfaces. Canvas textures draw markings and labels; an RGBA `DataTexture` carries energy to the ground shader.
Two white directional lights cast shadows for table and robot/city. Per-light callbacks avoid duplicate shadow writes. Night Sky HDRI 001 becomes the visible background and PMREM environment lighting [5, 6].

Scene ownership. `src/main.js` creates the runtime, world, audio, skyline, robot, player and control panel. Its frame loop updates player input, audio, controls, skyline and robot before issuing one render. Mathematical helpers remain separate from scene side effects.

Technical Report: Audio Implementation

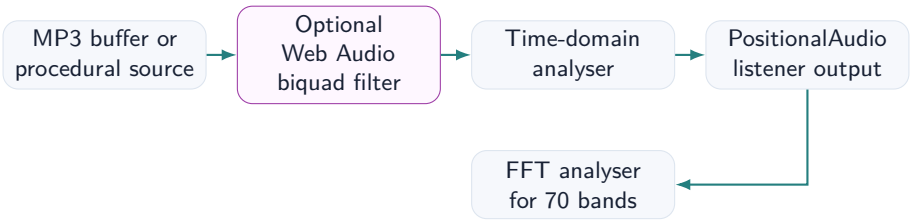


Figure 5: The processed signal drives audible output, waveform and spectrum.

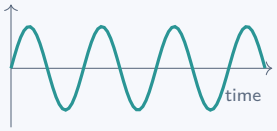
Source Lifecycle and Playback

The audio controller owns a Three.js `PositionalAudio`, one `AudioLoader`, four reusable `BiquadFilterNode` instances and the analysers. Selecting a source stops and disconnects the previous graph, clears the waveform, resets pitch, and increments a request version. Late callbacks from older asynchronous MP3 requests are ignored, which prevents an obsolete file from replacing the most recent choice.

Periodic signals use live oscillators. Noise is generated once into a looping buffer, and MP3 sources use decoded audio buffers. All sources are routed through the optional filter and time-domain analyser before the listener output. Pause lowers oscillator gain or pauses the buffer source without discarding the selected configuration.

Time Domain

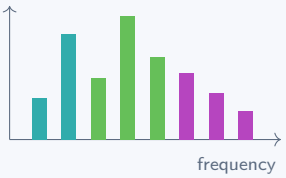
The waveform analyser reads 2048 post-filter float samples into one reused `BufferGeometry`. The display begins at the nearest rising zero crossing, while $\tanh(1.25x)$ keeps boosted samples inside the glass.



Question answered: how does amplitude change over time?

Frequency Domain

A 2048-point FFT uses a -90 to -8 dB range and low smoothing. Bins become seventy logarithmic bands from 20 Hz to 20 kHz; each controls one building.



Question answered: where is the signal's spectral energy?

Procedural Signals and Pitch

Periodic sources follow Fourier-series definitions:

$$x_{\text{square}}(t) \propto \sum_{k\ \text{odd}} \frac{\sin(k\omega_0 t)}{k},$$

$$x_{\text{triangle}}(t) \propto \sum_{k\ \text{odd}} (-1)^{(k-1)/2} \frac{\sin(k\omega_0 t)}{k^2}.$$

Sawtooth uses all harmonics; Rich uses eight weighted harmonics. Harmonics stop at 20 kHz and generated output is normalized to 0.35. Music maps to $r(p) = 0.5 + p$; oscillator pitch is logarithmic:

$$f(p) = 20\left(\frac{20000}{20}\right)^{p}\text{Hz}, \qquad 0 \le p \le 1.$$

Noise has no single fundamental and disables pitch.

Biquad Filters

Cutoff maps exponentially from 20 Hz to 20 kHz, resonance to $Q \in [0.1, 20]$, and peaking gain to $[-18, +18]$ dB.

Selecting low-pass, high-pass, band-pass or peaking rewires the same graph; selecting the active mode again bypasses it.

The red skyline curve uses equivalent magnitude equations. The FFT analyser receives the real filtered signal, so audible and measured effects remain aligned.

Technical Report: Spectrum and City Animations

Animation connects signal processing to the city. Each frame follows the same sequence: measure the signal, derive a visual target, smooth it and update reusable geometry or GPU state. The equations below are the mappings implemented by the application.

FFT Bands: From Decibels to Energy

Trigger. At 30 Hz, the analyser groups the 2048-point FFT into seventy logarithmic bands. For band i, containing n_i bins, the mean decibel value is normalized between -90 and -8 dB and slightly shaped:

$$\bar{d}_i = \frac{1}{n_i}\sum_{k=a_i}^{b_i} D_k, \qquad r_i = \operatorname{clamp}\left(\frac{\bar{d}_i - d_{\min}}{d_{\max} - d_{\min}}, 0, 1\right), \qquad e_i = r_i^{1.05}.$$

Meaning. $e_i \in [0,1]$ is the common input for the building, wireframe and ground animations, so all three views describe the same spectral measurement.

Source: skyline-visualizer.js:L738–L759

Buildings

Trigger. Each band energy sets the target Y scale of one instanced building:

$$s_i^* = s_0(h_i + 4e_i).$$

Attack and decay use different rates, $r = 200$ while rising and $r = 16$ while falling:

$$\alpha = 1 - e^{-r\Delta t}, \qquad s_i \leftarrow (1-\alpha)s_i + \alpha s_i^*.$$

Meaning. Height identifies where spectral energy is present without producing jitter. The live roof height also supplies the robot's vertical target.

Source: skyline-visualizer.js:L649–L678

Wireframe Skyline

Trigger. Line vertices reuse each building's smoothed scale. Each vertex stores a factor q_v; its height and recovered visual energy are

$$y_v = q_v s_i, \qquad \tilde{e}_i = \operatorname{clamp}\left(\frac{s_i/s_0 - h_i}{4}, 0, 1\right).$$

Brightness multiplies the logarithmic frequency color C_i:

$$b_i = 0.35 + 0.65\tilde{e}_i, \qquad C_v = b_i C_i.$$

Meaning. The wireframe preserves band identity while making active frequencies brighter and exactly matching the solid skyline height.

Source: skyline-visualizer.js:L702–L735

Ground Flow and Voronoi Surface

Trigger. Every band is written into the alpha channel of a 70-texel RGB energy texture. Energy is shaped and smoothed with the same exponential rule:

$$g_i = \operatorname{clamp}(e_i, 0, 1)^{1.45},$$
$$E^* = \min\left(0.80 \max_i g_i + 2\operatorname{avg}_i g_i, 1\right).$$

The overall value E is independently smoothed toward E^*.

Source: ground-flow.js:L14–L38

The fragment shader samples the texture by polar angle. Arc and radial masks constrain each colored fan, while a travelling wave modulates its intensity:

$$w(r,t) = \tfrac{1}{2} + \tfrac{1}{2}\sin(0.21r - 3.2t).$$

A procedural Voronoi field adds cell edges, Fresnel and specular glints.

Meaning. Energy appears to leave the source and travel toward the matching frequency region, while the floor remains subtly legible during silence.

Source: skyline-visualizer.js:L383–L443

Technical Report: Display and Control Animations

These animations expose the signal and the user's actions directly. Unlike the skyline, they operate on time-domain samples, analytical filter magnitudes or normalized input.

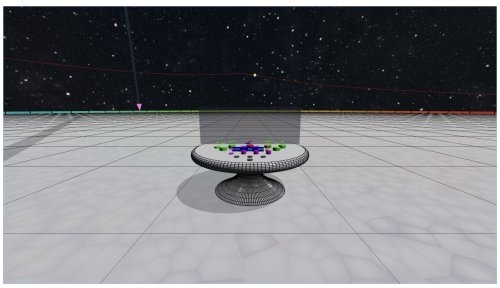


Figure 6: The red response curve and yellow cutoff marker update with the active filter.

How to Read It

The red curve follows magnitude across the same low-to-high arc as the buildings. Height predicts attenuation or emphasis, while the yellow point identifies cutoff. Knob motion updates sound and diagram together.

Filter Response

Trigger. Filter selection or knob motion recomputes one magnitude per skyline band. With $\rho_i = f_i/f_c$, $B_i = (1-\rho_i^2)^2$ and $A = 10^{g/40}$:

$$M_{\mathrm{LP}} = \frac{1}{\sqrt{B_i + (\rho_i/Q)^2}}, \quad M_{\mathrm{HP}} = \frac{\rho_i^2}{\sqrt{B_i + (\rho_i/Q)^2}},$$

$$M_{\mathrm{BP}} = \frac{\rho_i/Q}{\sqrt{B_i + (\rho_i/Q)^2}}, \quad M_{\mathrm{PK}} = \sqrt{\frac{B_i + (\rho_i A/Q)^2}{B_i + (\rho_i/(AQ))^2}}.$$

Magnitude becomes display height through

$$d_i = 20\log_{10} M_i, \qquad y_i = 0.6 + 29.4\,\mathrm{clamp}\left(\frac{d_i + 48}{60}, 0, 1\right).$$

Meaning. The curve predicts which skyline regions pass, fall or receive emphasis; the yellow point locates cutoff on the same logarithmic ordering.

Source: `spectrum.js:L22–L49`; `control-panel.js:L389–L417`

Live Waveform

Trigger. Every render frame reads 2048 post-filter samples. The displayed window starts at the nearest rising zero crossing n_0, stabilizing periodic shapes.

$$x_j = \mathrm{lerp}\left(x_{\min}, x_{\max}, \frac{j}{2047}\right),$$

$$y_j = y_c + \tanh\big(1.25u[n_0 + j]\big)\frac{H}{2}.$$

The hyperbolic tangent keeps boosted samples inside the glass without hard clipping.

Meaning. The red line reveals post-filter amplitude and periodic shape in the time domain, complementing the skyline's frequency-domain view.

Source: `audio-controller.js:L185–L230`; `audio-controller.js:L358–L378`; `waveform.js:L25–L33`

Knobs and Labels

Trigger. Vertical mouse motion changes normalized value p; $k = 0.005$ in free-pointer mode and $k = 0.01$ under pointer lock:

$$p' = \mathrm{clamp}(p + k\Delta y, 0, 1), \qquad \theta = \pi - (p - 0.5)\frac{3\pi}{2}.$$

The same value redraws the Hz, Q, dB or playback-rate label.

Meaning. Physical rotation and an exact number make parameter changes both spatial and measurable.

Source: `control-panel.js:L245–L280`; `control-panel.js:L419–L480`

Buttons

Trigger. Source, filter and pause selections toggle a pressed state. During the 150 ms transition,

$$\tau = \mathrm{clamp}\left(\frac{t - t_0}{150\,\mathrm{ms}}, 0, 1\right), \qquad y = y_0 - 0.06\tau,$$

with $1 - \tau$ used while releasing. The text sprite follows the same displacement.

Meaning. Persistent depth identifies the active source or filter without a separate menu; motion confirms that the click was accepted.

Source: `control-panel.js:L345–L385`

PERFORMANCE WITH PURPOSE Skyline analysis runs at 30 Hz while waveform and controls update at display refresh. Typed arrays, instancing, reusable buffers and one compact texture keep every animation responsive; each one explains sound or interaction rather than serving as decoration.

Hierarchical Robot and Animation

The pitch robot is the project's complex hierarchical model. It is assembled from nested Three.js groups and procedural meshes; animation is calculated in JavaScript. Each child inherits all upstream transformations, so the elbow follows the shoulder, the wrist follows both links and the stylus stays attached to the end effector.

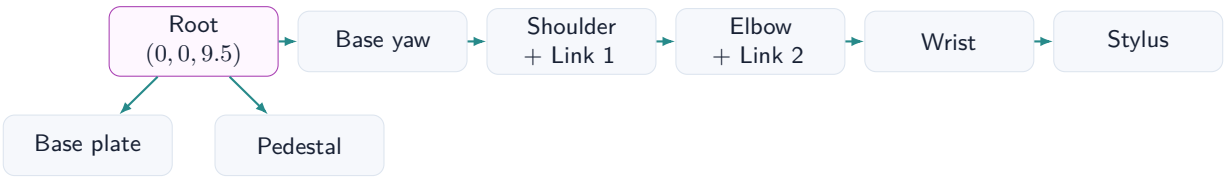


Figure 7: Parent-child transforms propagate from the root to the stylus.

Pitch-to-Target Mapping

For a periodic oscillator, normalized pitch is rounded to the nearest of the seventy skyline indices. The selected building contributes its world position and current roof height, so the target changes around the arc with pitch and vertically with FFT energy. MP3 playback changes rate rather than representing one frequency, and noise has no pitch; in both modes the robot returns to its inactive pose.

Source: `main.js:L96–L112`; `skyline-visualizer.js:L635–L642`

Two-Link Inverse Kinematics

The base yaw first aligns the arm with the target. In that vertical plane, distance d is clamped to the reachable interval for $L_1 = L_2 = 90$. The elbow follows the cosine rule,

$$\theta_2 = \cos^{-1}\left(\frac{d^2 - L_1^2 - L_2^2}{2L_1L_2}\right),$$

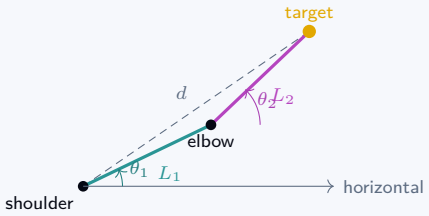
and the shoulder subtracts the triangle offset from the target direction:

$$\theta_1 = \operatorname{atan2}(y, x) - \operatorname{atan2}(L_2 \sin\theta_2, L_1 + L_2\cos\theta_2)\,.$$

The wrist applies $-(\theta_1 + \theta_2)$, preserving stylus orientation. Reach clamping prevents invalid inverse-cosine input and supplies a deterministic nearest pose.

Source: `two-link-kinematics.js:L3–L40`

Planar IK Geometry



The target and two fixed link lengths define the shoulder and elbow angles.

Active Tracking

While a periodic source is active, solved angles apply directly so the stylus maintains roof contact during abrupt building growth. Base movement controls decorative shoulder and elbow motor rings. The stylus glows bright magenta to communicate tracking.

Source: `pitch-robot-arm.js:L138–L186`

Inactive Motion

Without a periodic pitch, separate exponential damping rates return base, shoulder, elbow and wrist to a folded pose. Wrist compensation continues, motor rings stop and the emissive stylus fades, preserving mechanical character without implying a target.

Source: `pitch-robot-arm.js:L150–L186`

Why hierarchy is essential. Base yaw chooses the motion plane, shoulder and elbow solve reach, wrist compensation preserves the end effector, and the stylus inherits all parent transforms. The animation therefore exploits the model structure rather than moving independent parts by unrelated world-space values.

Conclusion and Future Work

Fourier City combines sound visualization, signal processing and 3D interaction into one coherent experience. The waveform shows the signal over time, the skyline distributes it across frequency, the filter response explains processing before and after it is heard, and the robot turns pitch into a physical target. Controls are embedded in the scene so the user can perform a transformation and immediately compare its audible and visual consequences.

The project demonstrates the required hierarchical model, lights, textures, interactions and original animations from the Interactive Graphics course brief [7]. More importantly, those requirements support a single educational intention: making the relationship between waveform, spectrum, pitch and filtering easier to explore.

Possible Future Improvements

- add band-stop and additional filter types with comparable response curves;
- support more generated signals, microphones and streamed audio sources;
- expand the city with more landmarks while preserving spectrum readability;
- make band count, frequency range, colors and smoothing user-configurable;
- improve spectral precision with selectable FFT sizes and measurement overlays;
- add keyboard-accessible controls, persistent help and higher-contrast labels;
- introduce saved presets for filters, sounds and visualization themes;
- optimize code splitting and lower-detail modes for weaker devices.

Assets and Attribution

The table and skyscraper OBJ files are original models created by the team in Blender. The scene uses *Night Sky HDRI 001* from ambientCG as a 4K EXR environment, distributed under Creative Commons CC0 1.0. [5, 6]. Three.js and Vite are used under their published open-source licenses. The repository is available at `https://github.com/Lorenzo-3/Fourier-City`.

References

[1] Three.js Authors. Three.js documentation. `https://threejs.org/docs/`, 2026. Accessed 19 June 2026.

[2] Mozilla Developer Network. Web audio api. `https://developer.mozilla.org/docs/Web/API/Web_Audio_API`, 2026. Accessed 19 June 2026.

[3] Vite Contributors. Vite guide. `https://vite.dev/guide/`, 2026. Accessed 19 June 2026.

[4] Blender Foundation. Blender. `https://www.blender.org/`, 2026. Accessed 19 June 2026.

[5] ambientCG. Night sky hdri 001. `https://ambientcg.com/view?id=NightSkyHDRI001`, 2024. 4K stitched HDR panorama; accessed 19 June 2026.

[6] ambientCG. License information. `https://docs.ambientcg.com/license/`, 2026. Creative Commons CC0 1.0 Universal; accessed 19 June 2026.

[7] Marco Schaerf. Interactive graphics: Project requirements. Course slides, Sapienza University of Rome, 2026.