Medusae Controls Guide (Detailed)

This document explains every control shown in the settings panel for the Medusae renderer and what each one changes visually.

Source locations used:
- src/data/settingsConfig.js (controls shown in the UI)
- packages/medusae/src/defaults.js (default values)
- packages/medusae/src/Medusae.jsx (actual behavior in shader + cursor follow logic)

Important distinction:
- Some controls change how far things move (amplitude/strength).
- Some controls change how fast things move (frequency/speed/drag response).
- Some controls change shape only (width/height/size/color).

CURSOR CONTROLS

1) Hover Radius (cursor.radius)
What it does:
- Sets the radius of the cursor jitter orbit used when hovering. The cursor target is not perfectly still; it is animated with a small wobble around the actual pointer.
Visual effect:
- Higher values make the interaction zone wobble farther around the cursor.
- The particle field appears to swirl around a larger moving target.
What it does NOT do:
- It does not directly increase particle size.
- It does not directly increase response speed.

2) Hover Strength (cursor.strength)
What it does:
- Multiplies the jitter offset added around the cursor target.
Visual effect:
- Higher values make the hover motion more dramatic and energetic.
- The cursor influence point moves farther from the pointer during the wobble animation.
When to increase:
- If you want more expressive cursor motion without changing smoothing.

3) Drag Factor (cursor.dragFactor)
What it does:
- Controls how quickly the internal mouse uniform (uMouse) catches up to the target cursor position using interpolation.
- The code effectively does: current += (target - current) * dragFactor
Visual effect:
- Higher values = faster, snappier cursor tracking.
- Lower values = smoother, laggier, floatier follow behavior.
This is the main ?speed with cursor? control.
Typical tuning guidance:
- 0.01 to 0.02 = soft / delayed

- 0.03 to 0.06 = responsive
- >0.08 = quite snappy and may feel less fluid

HALO CONTROLS

The halo is the jellyfish-like ring around the cursor that pushes and activates particles.
These settings mainly affect the ring shape and surrounding oscillation.

4) Outer Osc Frequency (halo.outerOscFrequency)
What it does:
- Sets the speed of the outer oscillation wave over time.
Visual effect:
- Higher value = faster pulsing/rippling motion in the outer region.
Tradeoff:
- Too high can look noisy or twitchy compared with the softer ?breathing? feel.

5) Outer Osc Strength (halo.outerOscAmplitude)
What it does:
- Sets how much the outer oscillation displaces particles in the outer region.
Visual effect:
- Higher value = stronger ripple push outside the main halo rim.
Tradeoff:
- Too high can overpower the halo rim effect and reduce the subtle jellyfish feel.

6) Outer Osc Jitter Strength (halo.outerOscJitterStrength)
Status:
- Present in the UI and defaults, but currently NOT wired into the renderer logic in packages/medusae/src/Medusae.jsx.
What that means:
- Changing this slider currently has no visible effect.

7) Outer Osc Jitter Speed (halo.outerOscJitterSpeed)
Status:
- Present in the UI and defaults, but currently NOT wired into the renderer logic in packages/medusae/src/Medusae.jsx.
What that means:
- Changing this slider currently has no visible effect.

8) Radius Base (halo.radiusBase)
What it does:
- Sets the base radius of the halo ring around the cursor.
Visual effect:
- Higher value = larger halo ring (the active band forms farther from the cursor center).
Use case:
- Increase this when you want a wider interaction footprint without making it stronger.

9) Radius Amplitude (halo.radiusAmplitude)
What it does:

- Controls how much the halo radius expands/contracts with the breathing cycle.
Visual effect:
- Higher value = more obvious breathing / pulsing size change.
Tradeoff:
- Too high can make the ring feel unstable rather than organic.

10) Shape Amplitude (halo.shapeAmplitude)
What it does:
- Adds irregular/noise-based deformation to the halo edge.
Visual effect:
- Higher value = rougher, less circular halo shape.
- Lower value = smoother, cleaner ring.
Use case:
- Raise it for a more alive / organic membrane look.

11) Rim Width (halo.rimWidth)
What it does:
- Controls the thickness/softness of the halo rim influence band via a smoothstep around the current halo radius.
Visual effect:
- Higher value = broader active rim (more particles affected around the ring).
- Lower value = tighter, thinner ring activation band.
Practical impact:
- Strongly affects how much of the scene lights up and grows near the halo.

12) Outer Start Offset (halo.outerStartOffset)
What it does:
- Defines where the outer oscillation influence begins relative to the base halo radius.
Visual effect:
- Higher value pushes the start of outer ripples farther away from the main ring.
Use case:
- Good for separating the core halo and outer ripple region.

13) Outer End Offset (halo.outerEndOffset)
What it does:
- Defines where outer oscillation influence reaches full effect (with smoothstep interpolation).
Visual effect:
- Changes the width of the transition area for outer ripples.
- Larger gap between start/end offsets = broader, softer outer ripple zone.

14) Halo Width (halo.scaleX)
What it does:
- Horizontal scaling of halo distance calculation.
Visual effect:
- Higher value stretches the halo wider left-right.
- Lower value compresses the halo horizontally.
Notes:

- This changes the shape of the influence region, not particle sprite shape.

15) Halo Height (halo.scaleY)
What it does:
- Vertical scaling of halo distance calculation.
Visual effect:
- Higher value stretches the halo taller up-down.
- Lower value compresses the halo vertically.
Tip:
- Adjust together with Halo Width to make the interaction feel round, oval, or flattened.

PARTICLE CONTROLS

These affect individual particle appearance, rotation behavior, and how strongly the field follows the cursor target.

16) Base Color (particles.colorBase)
What it does:
- The base/resting color used for particles when they are less activated by the halo (low vSize).
Visual effect:
- Dominates inactive areas of the particle field.

17) Color 1 (particles.colorOne)
18) Color 2 (particles.colorTwo)
19) Color 3 (particles.colorThree)
What they do:
- These three colors are blended in the fragment shader to create the active animated color gradients.
Visual effect:
- Active halo regions shift across this palette over time.
Notes:
- The labels are generic, but by default they are blue/red/yellow style accents.

20) Base Size (particles.baseSize)
What it does:
- Default particle size before halo activation boosts size.
Visual effect:
- Raises/lower overall particle presence even when far from the halo rim.
Tradeoff:
- Too large can make the scene feel dense and muddy.

21) Active Size (particles.activeSize)
What it does:
- Extra size contribution applied in active halo regions (scaled by rim influence).
Visual effect:
- Higher value makes particles ?bloom? larger when activated.
Use case:

- Increase to make the halo interaction more dramatic and readable.

22) Blob Width (particles.blobScaleX)
What it does:
- Horizontal scale of the particle sprite geometry after size is computed.
Visual effect:
- Higher value = wider particle blobs.
- Lower value = narrower particles.

23) Blob Height (particles.blobScaleY)
What it does:
- Vertical scale of the particle sprite geometry after size is computed.
Visual effect:
- Higher value = taller blobs.
- Lower value = flatter/squashed blobs.
Tip:
- Blob Width + Blob Height together define the particle aspect ratio.

24) Rotation Speed (particles.rotationSpeed)
What it does:
- Sets the time-driven speed of particle orientation jitter/rotation in the vertex shader.
Visual effect:
- Higher value = faster directional flutter/twist in the particles.
Notes:
- This affects rotational behavior, not cursor follow speed.

25) Rotation Jitter (particles.rotationJitter)
What it does:
- Controls the magnitude of angular jitter added perpendicular to the radial direction.
Visual effect:
- Higher value = more chaotic, wavy, less stable orientation.
- Lower value = cleaner alignment.
Tradeoff:
- Too much can make the particle field look noisy rather than fluid.

26) Cursor Follow Strength (particles.cursorFollowStrength)
What it does:
- Multiplies the cursor target position before it is applied to the internal mouse uniform target.
Visual effect:
- Higher value = particles react over a larger movement range (more displacement from cursor motion).
Important distinction:
- This mostly changes movement amount/range, not the smoothing rate.
- Pair with Drag Factor if you want stronger AND faster cursor interaction.

27) Oscillation Factor (particles.oscillationFactor)
What it does:

- Scales internal modulation used for particle rotation oscillation timing and intensity.
Visual effect:
- Higher value = more active, varied, lively particle motion.
- Lower value = calmer, steadier behavior.
Notes:
- This affects internal animation character, not the halo radius itself.

BACKGROUND CONTROL

28) Page Color (background.color)
What it does:
- Sets the page background color in src/App.jsx.
- Also sets the Three.js canvas background color in packages/medusae/src/Medusae.jsx.
Visual effect:
- Changes both the surrounding page and the canvas backdrop together.

PRACTICAL TUNING RECIPES

A) Faster cursor response (snappier)
- Increase Drag Factor first.
- Optionally increase Cursor Follow Strength a little.
- Keep Hover Strength moderate if motion becomes too chaotic.

B) Bigger interaction effect (more dramatic) without making it twitchy
- Increase Active Size.
- Increase Rim Width.
- Increase Cursor Follow Strength.
- Avoid increasing Drag Factor too much if you still want a floaty look.

C) Softer / more elegant jellyfish feel
- Lower Rotation Jitter.
- Use moderate Outer Osc Strength.
- Keep Shape Amplitude moderate.
- Use lower Drag Factor for smooth lag.

D) More chaotic / energetic effect
- Increase Rotation Speed and Rotation Jitter.
- Increase Oscillation Factor.
- Increase Outer Osc Frequency (carefully).
- Increase Hover Strength for stronger cursor wobble.

IMPLEMENTATION NOTE (Current Limitation)

Two Halo controls are currently exposed but not connected to the shader/runtime:
- halo.outerOscJitterStrength
- halo.outerOscJitterSpeed

If you want, these can be wired into the outer oscillation code so they add a second

jitter/noise layer to the outer ripple motion.

Generated on: 2026-02-26