Primordia

A 3D emergence laboratory: two-scale particle life, Barnes–Hut gravity and a piloted articulated creature

Interactive Graphics

Alessio Bandiera, Armando Coppola (1985878, 2003964)

Table of Contents

Overview

The idea

Overview

Complexity does not require complicated rules.

- Thousands of particles, **two force laws**, nothing else
- **Short-range "chemistry"** $\rightarrow$ cells, membranes, filaments, worms
- **Long-range gravity** $\rightarrow$ collapse, clumps, orbits, spiral arms
- The user never builds structures, only **edits the rules**

Satisfied constraints

- Everything procedural: no meshes, no images, no animation clips
- Everything live: no parameter change requires a reload
- The acceleration structures are on stage, not hidden

Environment

Overview

- **WebGL** through **three.js**, WebGL 2 when available
- `antialias`, high-performance GPU hint, sRGB output
- **Plain ES6 JavaScript**, no bundler nor transpiler
- Single-threaded, inside the animation-frame callback

What we did *not* write

Overview

One runtime dependency: three.js r128 (single `<script>` from a CDN)

renderer • scene graph • camera • primitives • `MeshStandardMaterial` • `InstancedMesh` • lights • `CanvasTexture` • `BufferGeometry` • math types

Everything else is ours

- No UI framework, no dat.GUI, no physics engine, no gl-matrix, no bundler
- Not even `OrbitControls`: our camera shares mouse buttons with the poke tools

Zero assets

- No `.obj`/`.gltf`/`.fbx`, no rigs, no keyframes, no images, no web fonts
- Geometry from parametric primitives, textures painted into a canvas at start-up

Credit: *particle life* / "Clusters" concept (Ventrella) • Barnes–Hut algorithm — implementations ours.

Table of Contents

Simulation engine

Architecture

Simulation engine

- 1 HTML + 1 CSS + **10 JS files**
- Load order *is* the dependency graph:
  `config` → `scene` → `particles` → `creature` → `grid` → `octree` → `simulation` → `camera` → `ui` → `main`

There is no `Particle` object, everything is an array

- Parallel typed arrays: positions, velocities, type, size, mass
- Cache locality: the inner loop streams, it does not chase pointers
- Allocated once $\Longrightarrow$ **zero allocations per frame** $\Longrightarrow$ the GC never pauses the animation
- Predictable types: JIT emits unboxed floating-point code for the whole loop
- Same discipline for the octree: 13 typed arrays, not a tree of node objects

Two scales, two data structures

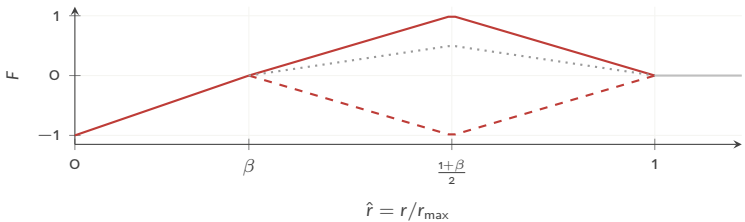
Simulation engine

	Chemistry	**Gravity**
Range	short, hard cut-off	infinite
Depends on	**type** (matrix)	**mass**
Naive cost	$O(N^2)$	$O(N^2)$
Structure	**uniform grid**	**Barnes–Hut octree**
Why valid	compact support	distant cluster $\approx$ its centre of mass

This pairing is the engineering core of the project.
Target: 1000–3000 mutually interacting particles at 60 Hz.

Chemistry: the force profile

Simulation engine



Force graph is piecewise linear, with three regimes:

1. $\hat{r} < \beta$: type-independent repulsion core
2. $\beta \leq \hat{r} < 1$: type-dependent interaction scaled by the matrix entry (the matrix is asymmetric)
3. $\hat{r} \geq 1$: the interaction is exactly zero beyond $r_{\max}$

Newton's third law is violated on purpose. Yellow chases purple, purple flees yellow: the pair self-propels. That asymmetry is the engine of every moving pattern.

Uniform grid

Simulation engine

- Cell size $=$ interaction radius $\Longrightarrow$ search only the **27 neighbouring cells**
- Rebuilt from scratch **every frame** by a **counting sort** for placing particles of the same cell **contiguous in memory**
- Build $O(N + \text{cells})$ allocation-free

At $N = 3000$

$9 \cdot 10^6$ pair tests per frame $\longrightarrow$ a few tens of thousands.
The HUD reports the pairs actually evaluated, live.

Gravity: Barnes–Hut octree

Simulation engine

- Pooled arrays indexed by node id, sized once
- Rebuilt every frame with **zero allocations**, children created lazily
- Guard on cell size: coincident bodies cannot subdivide forever
- **Explicit stack**, not recursion: bounded depth, no call frames
- Softening stops close encounters from slingshotting particles to infinity when $1/r^2 \to \infty$

θ is the algorithm's dial

θ	behaviour	cost
$\to$ 0	exact, opens all	$O(N^2)$
0.8	default	$O(N \log N)$
1.6	visible artefacts	fastest

The HUD node count makes the accuracy/speed trade-off legible in real time.

Integration, stability, boundaries

Simulation engine

Semi-implicit (symplectic) Euler, updating velocity first, then position based on the former

- Friction as a **half-life**, not a coefficient: framerate-independent and physically meaningful
- Time-step clamp: returning from a background tab costs slow motion, not an explosion
- Velocity clamp as the last line of defence

Two boundary modes, toggled live

- Reflecting walls with restitution
- Toroidal wrap: positions *and forces* wrap
- Gravity deliberately not wrapped

Table of Contents

Creature and visualisation

The Grazer: hierarchical modelling

Creature and visualisation

`root` $\rightarrow$ `hip` $\times 4$ $\rightarrow$ `thighPivot` $\rightarrow$ `knee` $\rightarrow$ `shinPivot` $\rightarrow$ `foot`

- Additional hierarchies for face $\rightarrow$ eye $\rightarrow$ pupil/glints, and antennae $\rightarrow$ stalk/bulb
- Cylinders for the limbs are translated **once at build time**, so the **pivot sits at the origin**
- Segments cannot come apart: the hierarchy makes separation unrepresentable

Procedural gait, and a creature that is physical

Creature and visualisation

No keyframes anywhere — a closed-form function of a phase accumulator

- Four legs, constant offsets a quarter-cycle apart to simulate a coordinated walk
- Cycle speed tracks body speed
- Secondary motion, all closed-form: antennae lean, stochastic blink, damped heading

Not a decorative overlay

- **Solid**: particles are projected out of its surface, impulse only if moving inward
- **Emits a field**: radial attract/repel *plus* a tangential swirl
- The swirl is why particles form an orbiting disc instead of a static ball

Visualising the invisible

Creature and visualisation

- **Octree wireframe**: large cells in empty space, tight subdivision around mass showing Barnes–Hut adaptivity
- **Gravity field streamlines**: up to 8000 samples, faint vector plus a travelling pulse;
  - field directions rebuilt every fourth frame (15 Hz)
  - pulse redrawn every frame (60 Hz)
- **Deforming space lattice**: 9^3 nodes displaced by gravity and by the user's poke

Performance

Creature and visualisation

Stage	Cost
Grid build	$O(N + \text{cells})$, allocation-free
Short-range forces	$O(N)$ expected
Octree build + traversal	$O(N \log N)$, allocation-free
Rendering	**one draw call** for N particles
Field / lattice	only when enabled

- 80 faces per sphere: 250k triangles at $N = 3000$
- No allocation in steady-state

Table of Contents

Interactions

Touch and strike

Interactions

Four drag tools: Orbit • Push • Pull • Warp

- Warp is swirl + collapse + axial compression
  - Swirl axis captured at drag start, and the whirlpool is always face-on
- A marker sphere shows the exact reach

Six one-shot impulses: Shockwave • Implode • Vortex • Warp • Jitter • Fold

Drag tools add a force, while impulses modify velocity.
Fold modifies position.

Piloting the Grazer

Interactions

- WASD + QE + boost; **force-based**, not positional
- Input builds an acceleration in the **creature's local frame**
- Same framerate-independent exponential damping as the particle friction
  - The result has weight: it accelerates, coasts, and **stirs the soup as it goes**
- Follow camera lerps at 5% per frame: an instant snap would jitter with the hover bob

The interaction matrix

Interactions

- $T \times T$ grid, rows act on columns; colour encodes sign and magnitude
- One click cycles a cell through five discrete values
- Read directly by the force kernel: **no apply button, no reseed**

Two clicks

Set one cell to $+1$ and its mirror to -1: type 0 chases, type 1 flees, the pair self-propels across the volume. Make the matrix symmetric instead: static clumps.
The user discovers asymmetry in ten seconds, without reading anything.

- **Nine presets**: each preset fixes the *regime*
- **Live reconfiguration**: particle count reallocates every buffer; world size *scales* positions instead of clipping them, so structures survive the resize

Table of Contents

Wrap-up

Limitations

Wrap-up

- **Single-threaded, CPU-bound** beyond about 3000 particles
- **Gravity is not periodic** under wrap
- **Creature's collision proxy is spherical**, not the visual silhouette that would take the legs into account
- **No foot-locking**: slight skating at high speed — fixing this issue would require inverse kinematics
- **Momentum is not conserved** — by design, and so there is nothing to validate against

Takeaway

Wrap-up

- Two simple laws at two scales produce cells, chains, filaments, orbits, spirals and swarms — **none of which exists anywhere in the source**
- Pooled typed arrays everywhere $\Longrightarrow$ zero steady-state allocation, no GC stutter
- Fully asset-free: 1 HTML + 1 CSS + 10 JS files, one dependency

Thank you for listening!
Any questions?