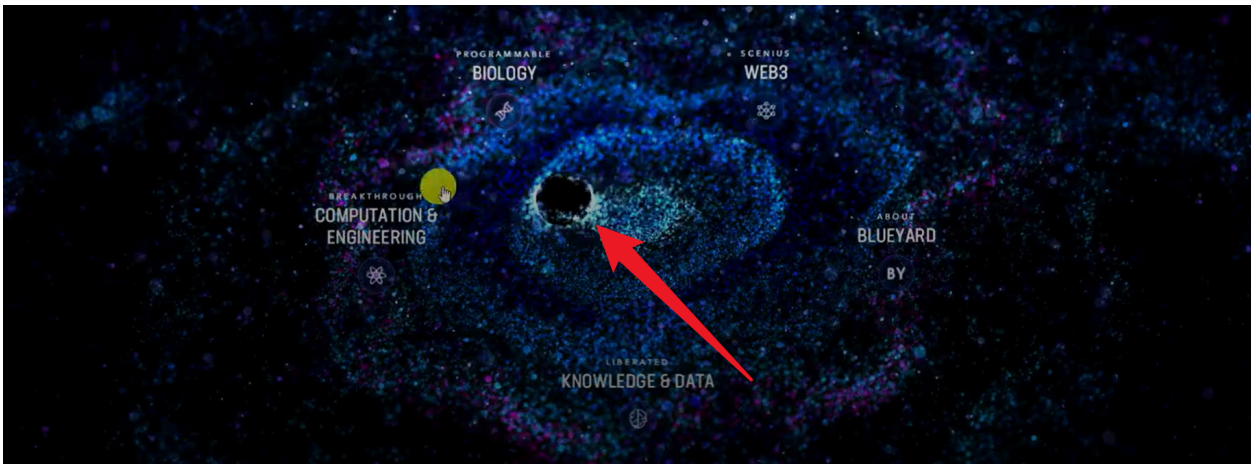
ThreeJS Trait Visualization: Solar System of People

Overview

The goal is to create an interactive, 3D visualization system that represents relationships and commonalities between individuals using a physics-based, solar system-like model. The system utilizes a configurable physics engine to simulate attraction and repulsion forces based on user attributes and preferences, visualized as a central node (the selected person) orbited by outer nodes (representing other people).
Frontend: Any framework that can showcase (in an app) and build an NPM package.
Canvas: Three.js and Spline for 3D rendering
Theme: Dark, neomorphic, glowy, and gooey aesthetic with neon purple (primary) and pink-red (secondary) with teal highlights. Nodes are the black holes with dummy particles swirling around them.



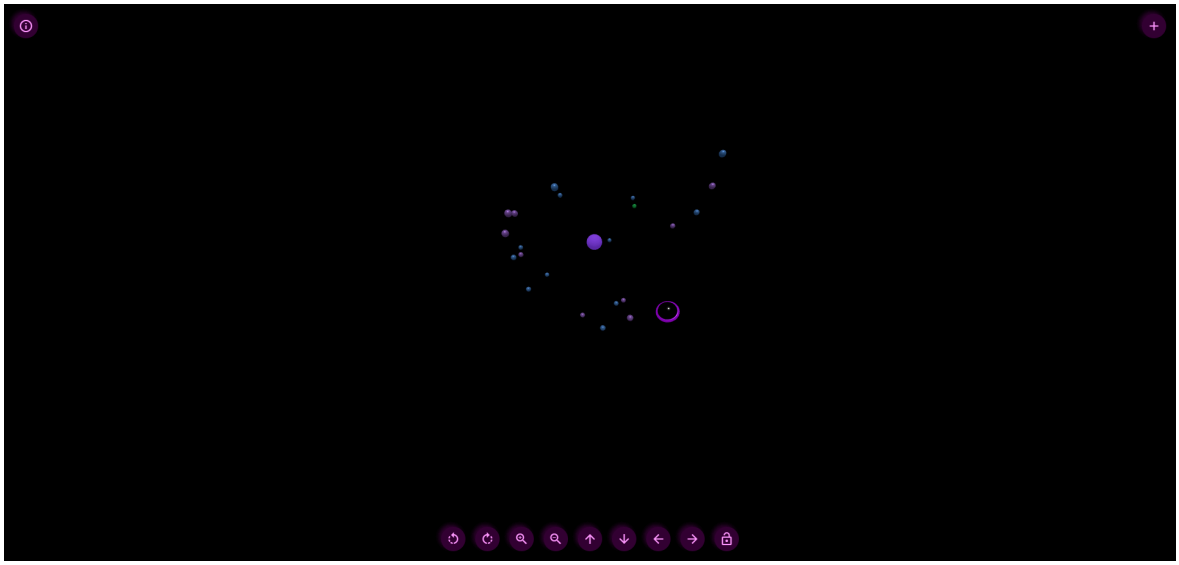
Requirements

1. **Make the UI look and interact like this, but with a Purple/Teal color theme instead of Blue/Purple.**
2. **Physics must be set up so that outside nodes with similar attributes to the central nodes' preferences gravitate to a closer position and vice versa. Also, outside nodes with similar attributes to each other gravitate to a closer position, too, and vice versa.**

The project is partially complete, and the task is to finalize it by addressing issues with the physics engine (ensuring accurate force calculations for multi-dimensional attributes) and implementing the specified theming. The system should dynamically adjust to user interactions, stabilize into an equilibrium, and provide a smooth, intuitive experience.

Current Work State

- **This is a video demo of the Current State. As you can see, the work just needs a polish, so please quote accordingly**
- This is what it currently looks like. It's off-theme and needs to be updated to look like this. This is the reference, and how it should look and feel. The reference shows a group of particles all with the theme colors, and the particles bend around the nodes, which resemble black holes.



- **Starting Point**: Upon project acceptance, a repository will be provided for you to work from (do not ask for access before being selected for the work). You are expected to push commits to the remote branch daily to demonstrate progress.
- **Issues to Address**:

    - The current physics model averages all attributes into a single score, which oversimplifies the visualization. Each attribute should contribute to the force calculations independently, treating attributes as spatial dimensions for conceptual clarity, with the system supporting up to 10 attributes.
    - Attraction and repulsion forces need to be inverted: higher compatibility (closer attribute-preference match) results in stronger attraction to the central node, and higher similarity between outer nodes results in weaker repulsion.
    - Theming is incomplete; adopt a dark, neomorphic design with neon purple (#C300FF) as the primary color and pink-red (#FF3366) as the secondary color.
- **Example of Desired Behavior**:
    - If the central node's preferences are `[100, 100, 100]` and an outer node's attributes are changed to `[100, 100, 100]`, the outer node should move closer to the central node, potentially colliding and "sticking" like magnetic marbles.
    - If two outer nodes have attributes `[77, 100, 77]`, they should cluster together due to reduced repulsion, orbiting closer to each other while influenced by the central node.

Objective

Create a 3D visualization system as a Library Component, with a showcase Library App, that:

- Represents a selected person as a central node and other people as outer nodes in a 3D space.
- Uses a physics engine to simulate forces:
    - **Attraction**: Between the central node's preferences and outer nodes' attributes, where higher compatibility results in stronger attraction to the center.
    - **Repulsion**: Between outer nodes, where greater dissimilarity results in stronger repulsion, and similar nodes cluster together.
- Supports at least 3 attributes initially (for visualization simplicity) and up to 10 attributes, with each attribute contributing to force calculations as if it were a spatial dimension.
- Ensures nodes settle into a stable, orbit-like equilibrium unless disturbed by user interactions (e.g., dragging nodes).

- Allows dynamic reconfiguration of parameters (e.g., number of nodes, attributes) and central node selection.
- Features a dark, neomorphic UI with neon purple (primary) and pink-red (secondary) colors, inspired by https://www.blueyard.com/ but with the specified color scheme.

The visualization should be smooth, interactive, and natural, resembling a solar system where nodes orbit the central node based on compatibility and similarity. Implementation details provided are suggestions to clarify requirements, and developers may propose alternative approaches if justified.

Key Features

1. Node and Attributes Setup

- **Nodes**:
    - **Central Node (Selected Person)**:
        - Represents the selected person with preferences (numerical values, default: 10 attributes, range 0–100).
        - Visualized as a larger, glowing neon purple sphere (radius: 2 units).
        - Positioned at the center of the 3D space.
    - **Outer Nodes (Other People)**:
        - Represent other individuals with attributes (numerical values, default: 10 attributes, range 0–100).
        - Visualized as smaller pink-red spheres (radius: 0.5–1 units) with customizable sizes/colors.
        - Initially positioned randomly in 3D space (±50 units).
- **Attributes and Preferences**:
    - Each node has a list of attributes (e.g., `[45, 72, 19, ...]` for outer nodes, representing traits like intelligence, creativity, empathy).
    - The central node has preferences (e.g., `[50, 70, 30, ...]`), compared to the outer nodes' attributes.
    - Start with 3 attributes for visualization simplicity (mapping to x, y, z for conceptual clarity), but ensure the system supports up to 10 attributes, with forces calculated per attribute. Future scalability to more attributes is desired.
    - Attributes and preferences are editable via a UI for testing.
- **Velocity**: Each node has a velocity vector updated based on forces, damped to prevent endless oscillation.

2. Force Mechanics

The physics engine drives the visualization by calculating forces based on attribute and preference comparisons. Each attribute contributes to forces independently, as if it were a spatial dimension, with the system supporting up to 10 attributes.

- **Attraction Force (Central to Outer Nodes)**:
  - **Formula**:
  - `F_{\text{attraction}} = k \times \text{Compatibility} \times \frac{\vec{d}}{|\vec{d}|^2}`
  - Where:
    - (k): Attraction constant (configurable).
    - `\text{Compatibility} = 1 - \frac{\text{Sum of absolute differences between preferences and attributes}}{\text{Max possible difference}}`.
    - `\vec{d}`: Vector from the outer node to the central node.
    - Max possible difference normalizes the score (e.g., for 10 attributes with range 0–100, max difference = `10 \times 100 = 1000`).
  - **Behavior**:
    - Higher compatibility (smaller difference) results in stronger attraction, pulling the outer node closer to the central node.
    - Example: For 3 attributes, central node preferences `[50, 50, 50]`, outer node attributes `[50, 50, 50]`, compatibility = 1, so attraction is strongest, pulling the node close.
    - Example: For attributes `[0, 0, 0]`, compatibility = `1 - \frac{150}{300} = 0.5`, resulting in weaker attraction and a farther orbit.
    - For 10 attributes, calculate differences per attribute and sum forces, ensuring scalability.
- **Repulsion Force (Between Outer Nodes)**:
  - **Formula**:
  - `F_{\text{repulsion}} = -k_{\text{rep}} \times (1 - \text{Similarity}) \times \frac{\vec{d}}{|\vec{d}|^2}`
  - Where:
    - `k_{\text{rep}}`: Repulsion constant (configurable).
    - `\text{Similarity} = 1 - \frac{\text{Sum of absolute differences between attributes}}{\text{Max possible difference}}`.
    - `\vec{d}`: Vector between two outer nodes.
  - **Behavior**:
    - Greater dissimilarity (larger attribute differences) results in stronger repulsion.
    - Higher similarity results in weaker repulsion, allowing nodes to cluster.

        - Example: For nodes with attributes `[77, 100, 77]` and `[77, 100, 77]`, similarity = 1, so repulsion is minimal, and they cluster.
        - Example: For `[77, 100, 77]` and `[0, 0, 0]`, similarity ≈ 0.153, so repulsion is stronger, pushing them apart.
- **Multi-Dimensional Attributes**:
    - Each attribute contributes to force calculations as a conceptual spatial dimension. For 3D visualization, start with 3 attributes for clarity (e.g., mapping to x, y, z forces), but ensure the system supports 10 attributes by summing forces across all attributes.
    - Example: For 10 attributes, calculate compatibility/similarity per attribute and aggregate forces into a 3D vector for visualization.
    - The system must remain stable and follow the same force rules for 3 to 10 attributes, with scalability for future expansion.
- **Equilibrium and Damping**:
    - Nodes settle into a stable, orbit-like configuration unless disturbed.
    - Apply damping (e.g., `\text{Velocity} = \text{Velocity} \times 0.95`) to stabilize the system.
    - Forces follow an inverse-square law for realistic behavior.

3. Interaction

- **Use a 3D galaxy model to make the motion effect (shown below)**:
    - The way the background particles flow over the surface of the black ball as it follows the cursor is achieved using Spline combined with JavaScript event listeners.
    - You can preview a similar setup here: https://app.spline.design/community/file/5ee9620e-2322-4dbd-a48c-9b8feb0f792f.
    - It uses a single object and minimal real-time computation, which ensures smooth performance.
    - That interaction should not require the physics engine. You just need the animation. The physics engine is only controlling what we are calling 'nodes'; the background particles are just there for the Spline interaction/animation that you will build. Any movement, including that which is governed by physics, is the same thing in the eyes of the interaction (a vector and a position), so the physics engine won't be directly used for this interaction.

- **Hover Tooltips**:
    - Display a node's attributes in a neomorphic tooltip when hovered, using Three.js raycasting.
- **Node Dragging**:
    - Allow users to click and drag nodes to disrupt equilibrium, with real-time force recalculation.
- **Central Node Selection**:
    - Provide a dropdown menu (not 3D clicking) to select the central node, updating preferences and forces.
- **Camera Controls**:
    - Use Three.js OrbitControls for panning, zooming, and rotating.

4. Real-Time Simulation

- Nodes move smoothly at 60 FPS, stabilizing into an equilibrium where:
    - Compatible nodes orbit closer to the central node.
    - Similar outer nodes cluster together.
- Example: For central node preferences `[80, 80, 80]` and outer nodes `[80, 80, 80]`, `[50, 50, 50]`, `[20, 20, 20]`, the first orbits closest, the second at a medium distance, and the third farthest. Similar nodes (e.g., `[50, 50, 50]` and `[55, 55, 55]`) cluster in their orbit.

5. Adjustability

- **Configurable Parameters**:

    - Number of outer nodes (1–20).
    - Number of preferences/attributes (3–10, default: 10).
    - Value of preferences/attributes (0-100, default: 50)
    - Force constants (attraction, repulsion, damping).
    - Node colors and sizes.
    - Lock/unlock cam.
- **Dynamic Updates**:
    - The visualization updates in real-time when parameters change.
    - Example: Adding nodes or changing attributes triggers a force recalculation and a new equilibrium.

6. Theming

- **The reference shows a bunch of particles all with the theme colors and the particles bending around the nodes, which look kinda like black holes.**
- **Visual Style**:
    - Dark, neomorphic, glowy, gooey aesthetic.
    - Primary color: Neon purple (#C300FF).
    - Secondary color: Pink-red (#FF3366).
    - Dramatic lighting with glowing nodes and ambient effects.
- **UI Elements**:
    - Neomorphic dropdowns and sliders for parameter controls.
    - Reference: Adapt https://www.blueyard.com/ aesthetic, replacing neon blue/purple with neon purple/pink-red.
- **Example**:
    - Central node: Large, pulsating neon purple sphere.
    - Outer nodes: Smaller pink-red spheres with varying glow intensity based on compatibility.
    - Background: Dark gradient (#1A1A1A to #2A2A2A) with star-like particles.

Implementation Plan (Suggestions)

The following are suggested steps to clarify requirements; alternative approaches are welcome with justification.

Step 1: Environment Setup

- **Framework**: Any frontend with Three.js.
- **Components**:
    - Develop a Library Component for the visualization.
    - Create a showcase Library App to demonstrate the component.
- **Setup**:
    - Initialize a Three.js scene with a dark background.
    - Add ambient and directional lighting.
    - Create a UI for parameter controls and node selection.

Step 2: Create Nodes

- **Central Node**: Large neon purple sphere, editable preferences.
- **Outer Nodes**: Smaller pink-red spheres, editable attributes.
- **Mock Data**: Store in `mock-data.ts`, use RxJS Observables with 1000ms delay.

Step 3: Physics for Configurable Forces

- **Attraction**: Calculate compatibility per attribute, summing forces for 3–10 attributes.
- **Repulsion**: Calculate similarity between outer nodes, applying inverse-square forces.
- **Damping**: Reduce velocity by 5% per frame.
- **Implementation**: Use a `PhysicsService.ts` for force calculations.

Step 4: User Interaction

- **Hover Tooltips**: Use raycasting for neomorphic tooltips.
- **Node Dragging**: Implement Three.js `DragControls`.
- **Central Node Selection**: dropdown component.
- **Parameter Controls**: forms for dynamic updates.

Step 5: Animation Loop

- Calculate forces, update positions, apply damping, and render at 60 FPS.
- Optimize for 20 nodes using spatial partitioning if needed.

Technical Requirements

- **Frameworks**: Three.js, Spline, OrbitControls (optional).
- **Languages**: TypeScript (strict typing), JavaScript (ES6+), HTML/CSS.
- **Libraries**: Open-source only, no API keys.
- **Code Structure**:
    - Strict typing in TypeScript.
    - Separate interfaces (e.g., `node.interface.ts`).
    - Mock data in `mock-data.ts` with RxJS Observables (1000ms delay).
    - Logic in services (e.g., `PhysicsService`, `NodeService`).
- **Repository**:
    - Work from the provided repository (shared upon project acceptance).
    - Push commits to the remote branch daily to demonstrate progress.
- **Delivery**:
    - Library Component and showcase Library App.
    - Prepare for future NPM package publication.

Deliverables

- **Code**:
    - Library Component for the visualization.
    - Showcase Library App demonstrating the component.
    - Organized codebase in the provided repository with strict typing, separate interface files, and mock data.
- **Features**:
    - Hover tooltips, draggable nodes, dropdown for central node selection.
    - Adjustable parameters (nodes, attributes, forces).
    - Smooth, equilibrium-based animations with neomorphic theming.
- **Progress Updates**:
    - Daily commits to the remote branch.
    - Weekly 30-minute video updates (via iTop Screen Recorder) every Monday and on-demand, viewable on Fiverr without downloading, with timestamps for demonstrated requirements.

Checklist for Video Signoff Requirements

Visual and Interactive Features

- **Node Representation**: Central node is neon purple, larger, glowy; outer nodes are pink-red, customizable (Timestamp: [TBD]).
- **3D Space Setup**: Random initial node positions, central node centered (Timestamp: [TBD]).
- **Forces and Movement**: Attraction based on compatibility, repulsion based on dissimilarity, equilibrium with damping (Timestamp: [TBD]).
- **Hover Tooltips**: Show node attributes in neomorphic tooltip (Timestamp: [TBD]).
- **Node Dragging**: Draggable nodes with real-time force updates (Timestamp: [TBD]).
- **Dropdown for Central Node Selection**: Updates central node and forces dynamically (Timestamp: [TBD]).
- **Smooth Animations**: 60 FPS, no jittering (Timestamp: [TBD]).

Configurable Parameters

- **Adjustable Node Count**: 1–20 nodes with dynamic updates (Timestamp: [TBD]).
- **Attribute Configuration**: 3–10 attributes, editable, real-time updates (Timestamp: [TBD]).
- **Force Settings**: Configurable attraction, repulsion, damping (Timestamp: [TBD]).

Technical and Functional Requirements

- **Responsive System**: Real-time updates, stable with 20 nodes (Timestamp: [TBD]).
- **3D Navigation**: Functional OrbitControls (Timestamp: [TBD]).
- **Stress Test Results**: Stable performance with 20 nodes, dynamic interactions, Chrome on stock pixel 7(Timestamp: [TBD]).

Theming and Design

- **Dark Neomorphic Theme**: Neon purple/pink-red, glowy, gooey, dramatic lighting (Timestamp: [T reference https://www.blueyard.com/ aesthetic, replacing neon blue/purple with neon purple/pink-red.
- **Example**:
    - Central node: Large, pulsating neon purple sphere.
    - Outer nodes: Smaller pink-red spheres with varying glow intensity based on compatibility.
    - Background: Dark gradient (#1A1A1A to #2A2A2A) with star-like particles.

Additional Examples of the Main Ask

- **Three Attributes Example**:
  - Central node: `[80, 60, 40]`.
  - Outer nodes: A `[80, 60, 40]`, B `[70, 50, 30]`, C `[20, 20, 20]`, D `[75, 55, 35]`.
  - **Behavior**:
    - Node A orbits closest (compatibility = 1).
    - Node B orbits at medium distance (compatibility ≈ 0.9).
    - Node C orbits farthest (compatibility ≈ 0.6).
    - Nodes B and D cluster due to high similarity (≈ 0.95).
    - Forces calculated per attribute and summed into 3D vector.
- **Ten Attributes Example**:
  - Central node: `[80, 60, 40, 50, 70, 30, 90, 20, 10, 100]`.
  - Outer node: `[80, 60, 40, 50, 70, 30, 90, 20, 10, 100]`.
  - **Behavior**:
    - Compatibility = 1, so the node orbits very close to the center.
    - For another node `[0, 0, 0, ..., 0]`, compatibility = 0, so it orbits far away.
    - Forces summed across 10 attributes, visualized in 3D space.
- **Interaction Example**:
  - User drags Node B, then changes central node to Node C.
  - **Behavior**:
    - Node B returns to equilibrium based on compatibility.
    - New central node triggers force recalculation, adjusting all orbits.
- **Parameter Change Example**:
  - Increase attributes from 3 to 10.
  - **Behavior**:
    - System recalculates forces for all attributes, maintaining stability.
    - Visualization remains in 3D, with forces aggregated into x, y, z components.

Prerequisites

- **End-to-End Solution**: Include all resources in the project quotation; no additional costs post-quote.
- **Blocking Issues**: Report before starting, as quotes are final.

- **Fiverr Account**: Must support Fiverr Custom Offers; no upfront payments.
- **Delivery**: Deliver only when the buyer agrees the project is complete.
- **Libraries**: Use open-source libraries only; no API keys.
- **Repository**: Daily pushes to the provided remote branch.

Progress Updates (Review Process)

- **Video Submission**:
  - 30-minute videos every Monday and on-demand via iTop Screen Recorder.
  - Include timestamps for demonstrated requirements (per checklist).
  - Viewable on Fiverr without downloading.
- **Repository Update**:
  - Daily commits to the provided remote branch.
  - Code reviews on the repository.
  - Stress tests to ensure <100ms latency with 20 nodes.
- **Stress Testing**:
  - Demonstrate stability with 20 nodes and dynamic interactions.

FAQ

- **UI/UX Reference**: Draw inspiration from: https://web.archive.org/web/20230828175256/https://www.blueyard.com/galaxy aesthetic with neon purple/pink-red colors. Use the Figma model (https://www.figma.com/design/UvNuBJc0aNsUaQAsdKWUu8/ForceDirectedGraphPersonCompatability?node-id=1591-349&t=5UKkkmHyrT2rqK4B-4) for force visualization concepts.
- **Theming**: Dark, neomorphic, glowy, gooey design with dramatic lighting.
- **Attributes/Preferences**: Start with 3 attributes/preferences for testing clarity, but ensure scalability to 10 attributes, with potential for more in the future. Also, use only 4 nodes for testing. This will make physics testing easier.