Great — since you're targeting the **web using Three.js**, here's exactly what you need to know about exporting animations involving **Squash & Stretch** from **Cinema 4D** to **.glb/.gltf** for reliable use in **Three.js**.

**🎯 Your Goal**

To export an animated 3D object (with squash/stretch behavior) from C4D into a .glb or .gltf format that plays correctly in **Three.js**.

**🚧 Problem**

Cinema 4D’s **Squash & Stretch deformer** is **not supported** natively in .glb — Three.js cannot interpret Cinema’s deformers, expressions, or procedural systems.

**✅ Solution Paths for Three.js**

**🥇 Option 1: Bake as Point Level Animation (PLA) → Alembic → Blender → .glb**

**Best for:** Complex deformations that can't be replicated with bones or morphs.

1. In Cinema 4D:
   * Select your mesh **with the deformer applied**.
   * Go to Character > Commands > Bake as Alembic.
     + Set proper frame range.
     + Make sure to enable "Include Deformations".
2. Import the .abc Alembic file into **Blender**.
3. In Blender:
   * The baked animation will appear as vertex animation.
   * Export as .glb:
     + File → Export → glTF 2.0 (.glb)
     + Enable Animation > Shape Keys and Apply Modifiers.
4. In Three.js:
   * Use GLTFLoader to load the model.
   * The animation will play automatically if handled correctly.

⚠️ PLA = large file sizes (every vertex on every frame), but it's accurate.

**🥈 Option 2: Use Morph Targets (Blend Shapes)**

**Best for:** Simple squash/stretch between two or more states.

1. Create your **default**, **squashed**, and **stretched** mesh versions in C4D.
2. Add a **Pose Morph tag** to the base mesh.
   * Add each target shape (squash and stretch).
   * Animate the morph sliders over time.
3. Export to .fbx or bring into Blender.
4. In Blender:
   * Check that shape keys exist and animate correctly.
   * Export as .glb with shape keys enabled.
5. In Three.js:
   * You can control morph weights manually or via animation mixers.

**🥉 Option 3: Rig the Object with Bones (Joint-Based Animation)**

**Best for:** Clean, efficient, reusable squash/stretch effects.

1. Replace the deformer with a simple rig:
   * Use joints to squash/stretch parts of the mesh.
   * Animate joint scales or transforms.
2. Bake the animation (if needed) and export as .fbx.
3. Import .fbx into Blender → Export as .glb.
4. Three.js supports bone animations natively via AnimationMixer.

✅ Best performance and file size. ❌ Slightly more setup.

**✅ Summary for Three.js Export**

| **Method** | **Web Perf** | **File Size** | **Fidelity** | **Difficulty** |
| --- | --- | --- | --- | --- |
| **Alembic → PLA** | ❌ Medium | ❌ Large | ✅ High | 🔧 Medium |
| **Morph Targets** | ✅ Great | ✅ Small | ✅ Medium | 🔧 Medium |
| **Bone Rigging** | ✅ Best | ✅ Small | ✅ High | 🔧 High |

**Bonus: C4D to Three.js .glb Export Tip**

Since Cinema 4D doesn’t export .glb natively:

* Use **FBX or Alembic → Blender → .glb** for best compatibility.
* Blender is your best middleman to clean up, test, and export animation for the web.

Let me know which of the three workflows you'd like help setting up — I can walk you through the exact steps based on your specific animation.

Thanks for the clarification — this changes things significantly.

**Key Point:**

You're using a **Voronoi Fracture simulation baked as Alembic** in **Cinema 4D**, then exporting to **GLB**.

**🔍 The Problem:**

**GLB/GLTF does *not* support Alembic-style vertex animations (a.k.a. point cache or per-frame mesh deformation)** natively. That includes:

* Voronoi fracture simulations
* MoGraph caches
* Cloth/hair simulations
* Any mesh that changes **topology** or **per-vertex positions** over time (as Alembic typically does)

This is why:

* You don’t see animations in the GLTF viewer or Three.js
* The GLB loads as static geometry only

**✅ What GLB *Does* Support:**

* **Skeletal/bone animations** (e.g. rigged characters)
* **Object-level transforms** (position, rotation, scale)
* **Morph targets / shape keys**

But not:

* Per-frame changing geometry (as in Alembic)

**✔️ Solutions for Web (Three.js):**

**1. Bake to Keyframes (Transform only)**

If the simulation can be approximated with only transform animations (like chunks flying apart), try converting it to animated object transforms (PLA or PSR animation in C4D), and export with baked keyframes.

**2. Bake to Morph Targets (if applicable)**

If you have a few mesh states (not hundreds of frames), consider baking those as morph targets. GLTF supports this.

**3. Use a Vertex Animation Texture (VAT)**

Three.js can support **vertex animation textures**, which store vertex positions per frame in a texture. You’d:

* Export the Alembic cache to a VAT using tools like **Houdini** or **Blender** with plugins
* Use a special **custom shader** in Three.js to decode and play back

See examples like:

* [VAT example in Three.js](https://threejs.org/examples/#webgl_animation_skinning_morph)

**4. Export Alembic → Play as Video (Alternative)**

If you're using the simulation just for visual flair, sometimes exporting it as a **pre-rendered video** or even using a **sprite sheet** might work for web display.

**✅ Best Path for You:**

If you need real-time object fracture in Three.js:

* Bake each fractured chunk as a separate object
* Animate them with keyframes (or use physics in Three.js)

If you want to export the exact simulation:

* Use VAT or pre-rendered media.

Would you like help creating a VAT-compatible export pipeline or baking your Voronoi sim into keyframed chunks in C4D?

Great question — converting a baked **Alembic (.abc) point cache** into something that can be exported in .glb (i.e. shape keys or object-level keyframes) requires a few specific steps in **Blender**. Here's a practical breakdown for both methods:

**🎯 Goal**

Make an imported Alembic animation exportable via .glb by converting:

* Vertex animation → **shape keys** (if topology is constant)
* Object motion → **object-level keyframes** (transform animation)

**🅰️ Option 1: Convert Alembic to Object Transform Keyframes**

✅ Best if the animation is just moving/scaling/rotating objects (e.g. squash/stretch)

**🔧 Steps:**

1. **Import the Alembic file**
   * File > Import > Alembic (.abc)
2. **Select the animated object**
3. **Go to frame 1**, press A in 3D view to select all relevant objects
4. In the timeline or 3D view:
   * Run Object > Animation > Bake Action
5. **In Bake Action settings**:
   * Check:
     + Only Selected
     + Visual Keying ✅
     + Clear Constraints ✅ (optional)
     + Overwrite Current Action
     + Frame Range (match your animation length)
6. Click **OK**

This creates keyframes on location, rotation, and scale — and **removes the Alembic modifier**.

1. Now export via File > Export > glTF 2.0 (.glb)
   * Make sure **Animations** is checked

**🅱️ Option 2: Convert Alembic to Shape Keys (Morph Targets)**

✅ Best for deformations where the object’s shape changes but topology remains constant (e.g. squash/stretch mesh deformation)

**🔧 Steps:**

1. **Import the Alembic file**
2. Scrub the timeline and identify key moments in the animation (e.g., neutral, squashed, stretched)
3. **At each key moment**:
   * Duplicate the mesh (Shift+D)
   * Move it to another collection or layer to keep things clean
   * This gives you "snapshots" of your mesh over time
4. **Go to the base (rest) mesh**
   * Add a **Basis shape key**: Properties > Object Data > Shape Keys > +
5. Add additional shape keys for each pose:
   * Click + again
   * Enter Edit Mode
   * Use Join as Shapes to import each duplicate mesh as a shape key
6. **Animate shape key values**:
   * In the **Dope Sheet > Shape Key Editor** or Timeline
   * Insert keyframes on shape key influence (hover over the value, press I)
7. Export using File > Export > glTF 2.0 (.glb)
   * Enable **Shape Keys** and **Animations**

**🟨 Notes:**

* **Alembic = baked mesh** — no bones or real-time deformation
* **GLB doesn’t support full vertex cache** → must be transformed into animation formats GLB understands
* For squash/stretch, **object transforms** or **morph targets** are ideal

**🧩 TL;DR**

| **Animation Type** | **Best Export Method** | **GLB-Compatible** |
| --- | --- | --- |
| Object transform | Bake Action | ✅ Yes |
| Vertex shape change | Shape Keys | ✅ Yes |
| Topology-changing sim | Not supported | ❌ No |

Let me know your specific squash/stretch setup — I can guide you on whether transforms or shape keys are more appropriate and give step-by-step Blender actions for it.