Metaball Madness: Look Development For A Shapeshifting, Implicit Surface Character On Pixar's *Elio*

CATHERINE LUO, TRENT CROW, FERNANDO DE GOES, and FERDI SCHEEPERS, Pixar Animation Studios, USA

How do you shade a liquid supercomputer created as an implicit surface? *Elio*'s OOOOO is Pixar's first character made and rigged entirely as a controllable series of metaball-like signed distance functions, rendered for interaction with a GLSL shader [Lykkegaard et al. 2025] in our animation software, *Presto*. Although this novel approach facilitates incredible animation, it does not provide a stable mesh for shading, rendering, and deformation purposes. OOOOO is made of a body and blobs, all capable of separating and merging at any time, filled with "nests" of moving circuits that are concentrated at the core. This talk addresses the accompanying set of challenges these factors created for look development past the model/rig stage, resulting in a per-frame process.

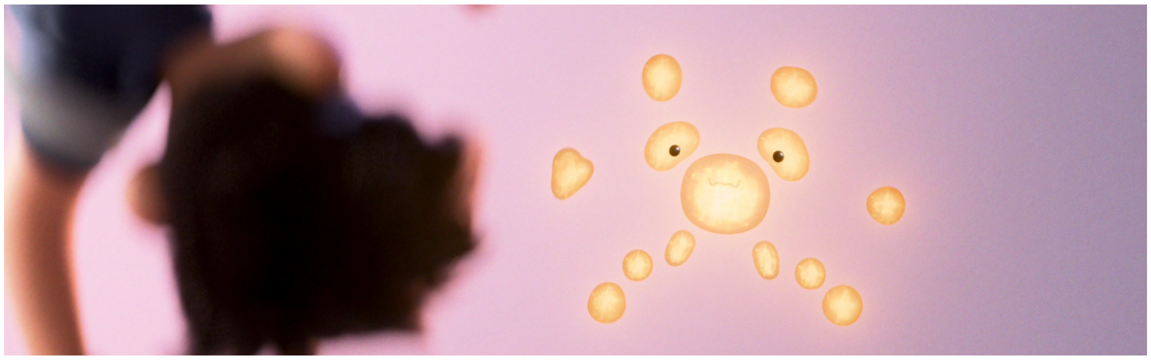
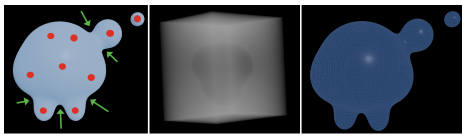


Fig. 1. *Elio*'s OOOOO has an implicit surface rig that can take many shapes ©Pixar

1 From Math to Mesh



Since the animated character is represented only as implicit functions, we need a mesh representation of those functions to integrate her into the rest of the pipeline so she can be shaded, lit, and rendered. To do this, we use the implicit functions as signed distance functions to define a density volume in Houdini, converting the GLSL code defining OOOOO into equivalent Houdini VEX code. This code defines the volume that is then converted into a polygonal mesh along its zero-density isosurface.

While this provides a mesh to use, it is topologically inconsistent frame to frame, which presents a few challenges. One is how to calculate motion blur, which uses vector offsets between corresponding mesh points. Without a consistent mesh, we instead use proxy geometry that is consistent between frames. We create spheres at the center of each implicit shape and then calculate their offsets between frames before transferring them to the closest points on the inconsistent mesh. This provides consistent vectors frame to frame that approximate the motion direction of the character.

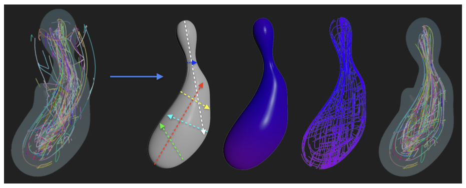
OOOOO's look depends on her circuit geometry correctly orienting to her newly meshed body as it moves, so without a consistent mesh nor standard rig, we instead use a modified version of our custom tool *Geomt Wrapper* [de Goes

Authors' Contact Information: Catherine Luo, catluo@pixar.com; Trent Crow, tcrow@pixar.com; Fernando de Goes, fernando@pixar.com; Ferdi Scheepers, ferdi@pixar.com, Pixar Animation Studios, USA.

and Martinez 2019]. Geomt Wrapper is an iterative method that wraps a source mesh to a target shape by alternating closest-point projections and a detail-preserving relax. For OOOOO, we introduce a new mode to this tool that replaces the relax solver with a fitting step, thus returning the best global affine transformation, mapping the source model to the latest projected points. By repeating these iterative steps, we get a deformation of the source model (the proxy mesh) that orients it towards the target shape (the VDB-generated posed mesh) free of any input correspondences.

2 Circuitry And Shapeshifting

OOOOO's circuit nest involves distance-maximized point pairs scattered on her surface that are run through a shortest path calculation, creating a dense grid of curves that are offset into her body and carved according to frame calculations. This initial system for OOOOO was developed on a stable body mesh with no blobs, and does not account for extreme shapeshifting poses. Because the system uses a static float value per curve to offset along the reverse normal of OOOOO's body, it does not compensate well for varying thickness, resulting in penetration of the body or lack of the core nest effect. To address this while keeping the integrity of the original look and system, we created the idea of a dynamic, ray distance multiplier (RDM). The RDM is created per frame by sending rays from each point on the body mesh inwards along the reverse normal, measuring distance to first hit. Outlier values are cut off, and the point data is then blurred across the mesh and normalized. After a nearpoint operation is used to transfer the RDM to the circuitry curves, it is then applied to the original offset per curve point, resulting in neat circuitry with a clear nest in all shapes (see inset).



3 Primvar Creation For Shading and Lighting

To meet the artistic direction of OOOOO, it is necessary to create primvars within Houdini to target our character's color, specular, etc., in specified areas. To make these primvars, we split the initial meshing process of OOOOO at the sparse volume stage into several parts: mouth, eyes, smooth body without face nor arms, and full resolution mesh with face and arms. Using these components, we complete a series of boolean and volume operations to create new isolated meshes of desired parts from OOOOO, which we can attach attributes to that are then transferred to her final render mesh. These are referenced in our shading software *Flow*, and used to create a much more appealing character shader.

References

Fernando de Goes and Alonso Martinez. 2019. Mesh wrap based on affine-invariant coordinates. In *ACM SIGGRAPH 2019 Talks (SIGGRAPH '19)*. Association for Computing Machinery, Article 4, 2 pages. doi:10.1145/3306307.3328162

Anna-Christine Lykkegaard, Andrew Butts, and Julian Teo. 2025. Metaball Madness - The Rigging Of An Implicit Surface Character. In *under review*.