AquaGen: Interactive Aquarium Generated From Text

Dingning Cao [1], **Yifan Kang** [1]

[1]Massachusetts Institute of Technology (MIT)

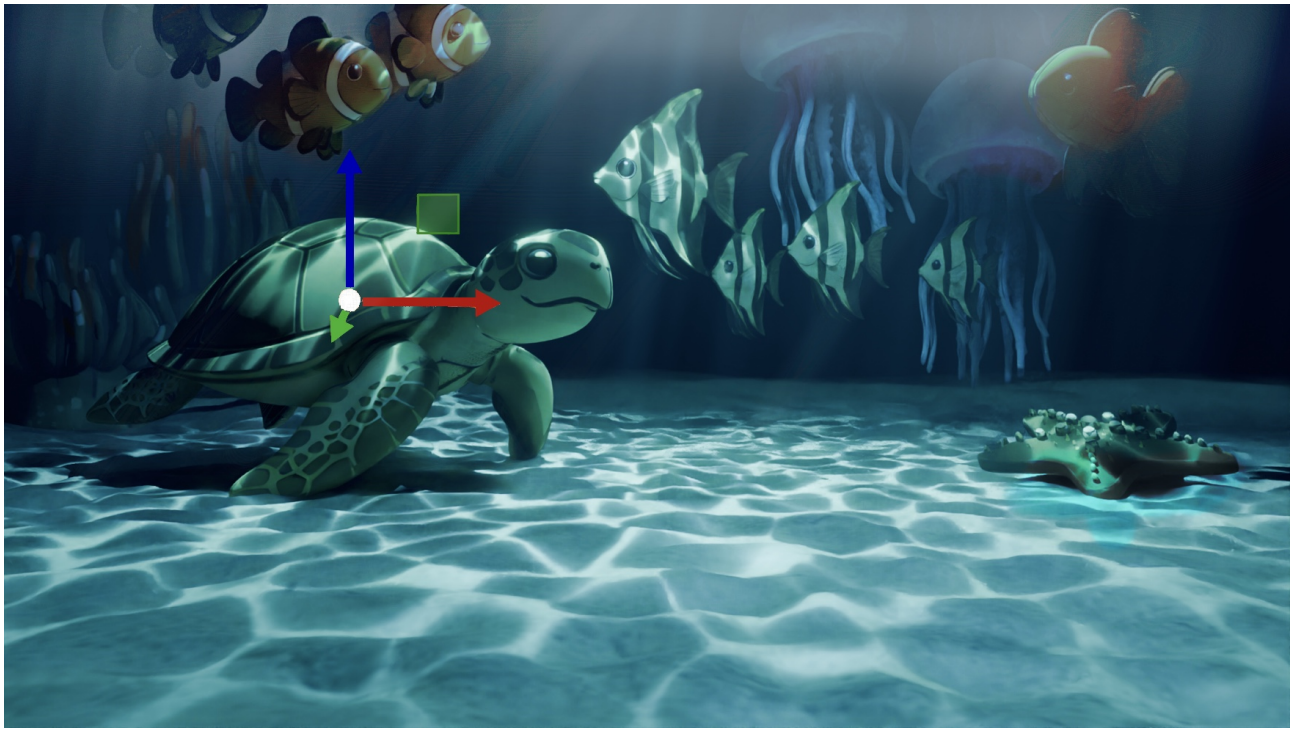

Fig. 1: Pre-rendered scene using sea creatures generated through our proposed pipeline

1. Motivation

We aim to create a highly customizable, interactive virtual aquarium where users can generate and animate any fish from a simple text or image prompt. This project merges generative modeling, physics-based animation, and real-time interaction into a unified creative experience. However, realizing this vision presents key challenges. Diffusion-based 3D generation models such as Hunyuan3D Lai et al. (2025) produce richly detailed assets, but their dense and unstructured meshes are difficult to animate or deform physically. Traditional hand-authored animations also lack physically realistic secondary motion—subtle movements like fin swaying or body jiggle—unless supplemented by computationally intensive physics simulations Zhang et al. (2020), which are often unsuitable for real-time applications. Additionally, standard 3D animation pipelines involve complex and specialized steps including modeling, rigging, and simulation, typically requiring a team of artists. Our project addresses these gaps by introducing a fully automated pipeline that empowers users to generate and animate fish models in an interactive scene with minimal technical overhead.

2. Related Work

Our project draws from a broad range of prior works in 3D content creation, meshing, simulation, and rendering. Hunyuan3D Lai et al. (2025) enables high-fidelity textured mesh generation from 2D images, while Nano-Banana Pro Google DeepMind (2025) pushes the boundary of generative models for text-to-image synthesis, offering a flexible foundation for user-driven asset creation. To convert these surface meshes into volumetric representations suitable for simulation, we consider Fast Tetrahedral Meshing in the Wild Hu et al. (2020), which produces high-quality tetrahedral meshes robustly from noisy or complex geometry. For dynamic deformation, Fast Complementary Dynamics via Skinning Eigenmodes Otman Benchekroun (2023) presents a reduced-space solver that enhances models with physically based secondary motion using a rotation-equivariant subspace formulation decoupled from mesh complexity. Finally, for scene immersion, Guardado's work on water caustics simulation demonstrates how light refraction can be used to create dynamic, visually compelling underwater effects, which can be adapted to enrich our aquarium environment.

3. Technical Work

3.1. Complementary Dynamics

Complementary dynamics provides a method to augment rigged animations with detailed elastodynamics. The displacement field can be described as:

$$u = u^r + u^c,$$

where u^r is the prescribed rigged motion. To simulate realistic physical effects, the system computes the *secondary motion* u^c as the deformation complementary to the motion imposed by the handle. Traditionally, u^c is obtained from a physics simulation, which can be expressed as an energy minimization problem:

$$u^c = \text{argmin}_{u^c} E(u^c + u^r + x_0) \quad \text{s.t. } J^T u^c = 0$$

where J is the rigged Jacobian. Here, the complementary constraint enforces that the physical motion must not be in the space of motion produced by the rig, $\text{Col}(J)$. Zhang et al. (2020) introduced a way of enforcing the constraints through Lagrange multipliers and minimizing the resulting energy using Newton's method, requiring the frequent solve of the following system:

$$\begin{bmatrix} H & J \\ J^T & 0 \end{bmatrix} \begin{bmatrix} du^c \\ \lambda \end{bmatrix} = \begin{bmatrix} -g \\ 0 \end{bmatrix}$$

where H is the elasto-dynamic Hessian matrix. Iteratively solving this system is too expensive for real-time applications because it scales with mesh resolution and the constraint matrix $J^T u^c = 0$ is typically dense.

Thus, we used the approach introduced in Otman Benchekroun (2023). secondary motion is approximated using a low-dimensional subspace of **skinning eigenmodes** B, such that: $u^c \approx Bz$ Here, z represents the coefficients in the reduced basis. The key principle of *complementarity* ensures that u^c is orthogonal to any deformation that the affine handle can cause. This is enforced implicitly by constructing B through a constrained generalized eigenvalue problem (GEVP). During each real-time frame, the system reads the current pose p, solves for the optimal secondary coefficients z,

and updates the fish model by combining both motion components. Furthermore, we introduced **secondary motion scale**, a value from 0 to 1, to control the strength of secondary motion we add to the model.

3.2. Model Generation Pipeline

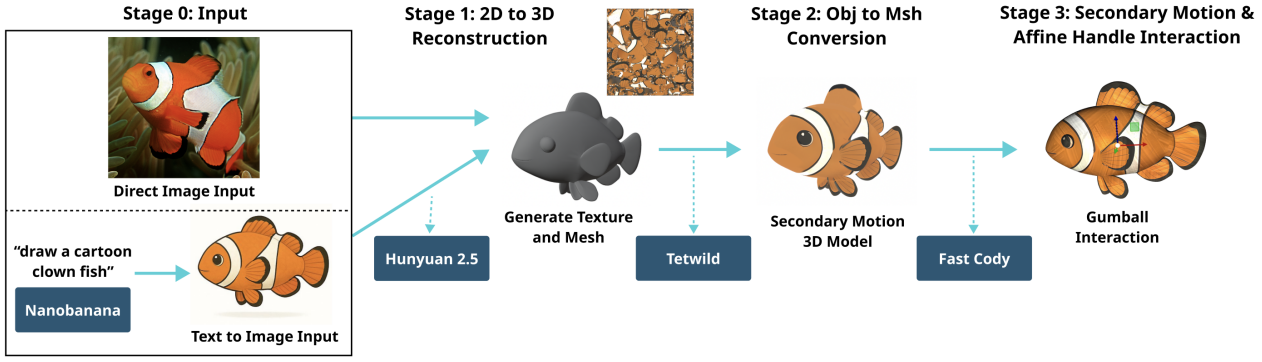

Fig. 2: The 4 stages of AquaGen Pipeline

Our pipeline enables customized fish animation through a four-stage process: (1) Users begin with either a natural image or a prompt, which is turned into a stylized fish image via Nano-Banana Pro Google DeepMind (2025). (2) The resulting image is processed by Hunyuan3D 2.5 Lai et al. (2025) to synthesize a textured 3D model. (3) The mesh is also converted into a tetrahedral format using TetWild. We lowered the envelope from $b/1000$ to $\mathbf{b/1500}$ and increased the ideal edge length from $b/20$ to $\mathbf{b/25}$, striking a balance between preserving geometric detail and avoiding excessive mesh density.(4) Finally, Fast Complementary Dynamics (Fast Cody) simulates realistic secondary motion on the tetrahedral mesh using a linear blend skinning–based eigenmode solver. Affine handles are attached to each fish model to allow direct user interaction, completing a streamlined pipeline for interactive, physics-based animation of generated assets.

3.3. Shader and Scene Setup

3.3.1. Underwater Lighting and Caustic Effects

Our fragment shader combines physically based shading, animated caustics, gradient backgrounds, and distance fog to approximate the appearance of an underwater environment. The base color is computed using a Blinn–Phong model:

$$I = I_a + I_d + I_s,$$

with ambient, diffuse, and specular components modulated by material coefficients (K_a, K_d, K_s).

3.3.2. Animated Caustics

Caustics are produced using a horizontal texture atlas of N frames. We animate by advancing the frame index and warping the UVs with drift and wobble:

```
int frame = int(floor(u_time * u_frameRate)) % u_numFrames;
vec2 baseUV = texcoordi * 2.0;
vec2 drift   = vec2(u_time*0.15, u_time*0.10);
vec2 wobble  = vec2(sin(u_time+v_worldPos.x)*0.02,
                    cos(u_time+v_worldPos.z)*0.02);
vec2 tiledUV = fract(baseUV + drift + wobble);

float frameWidth = 1.0 / float(u_numFrames);
vec2 uv = vec2(frame*frameWidth + tiledUV.x*frameWidth, tiledUV.y);
float C = texture(u_causticsAtlas, uv).r;
```

Bright caustic values are added to the material:

$$I_{\text{final}} = I_{\text{surf}} + 0.15\, C\, \mathbf{c}_{\text{tint}}.$$

3.3.3. Underwater Fog

We approximate depth attenuation using linear fog in eye space:

```
float d = length(position_eye);
float fog = clamp((d - 5.0) / (40.0 - 5.0), 0.0, 1.0);
finalColor.rgb = mix(finalColor.rgb, FOG_COLOR, fog);
```

3.3.4. Bubble Particle System

We generate underwater bubbles as a particle system using instanced sphere meshes. Each bubble is represented as a UV sphere mesh generated using spherical coordinates.

Bubbles spawn randomly within a horizontal range $[-4, 4]$ at ocean floor level $y = -0.5$, with each bubble assigned a random upward velocity $v \in [0.7s, 1.3s]$ where $s = 0.3$ units/second. When a bubble reaches height $y > 5.0$, it respawns at a new random position on the floor. Each frame, bubble positions are updated as $y_{t+1} = y_t + v \cdot \Delta t$, and the mesh vertices are translated accordingly.

4. Results

Live Demo Video: Click here to watch the live-captured demo video (Macbook 2020 Intel).

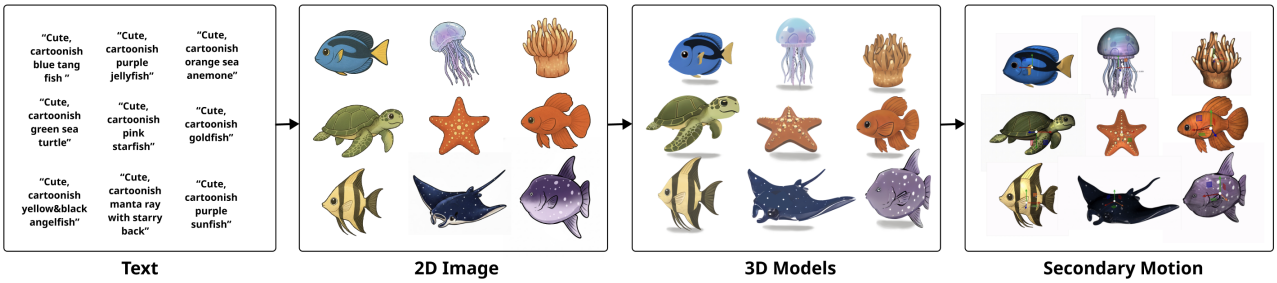

Fig. 3: Nine models that are created through the AquaGen pipeline: from text to interactive model

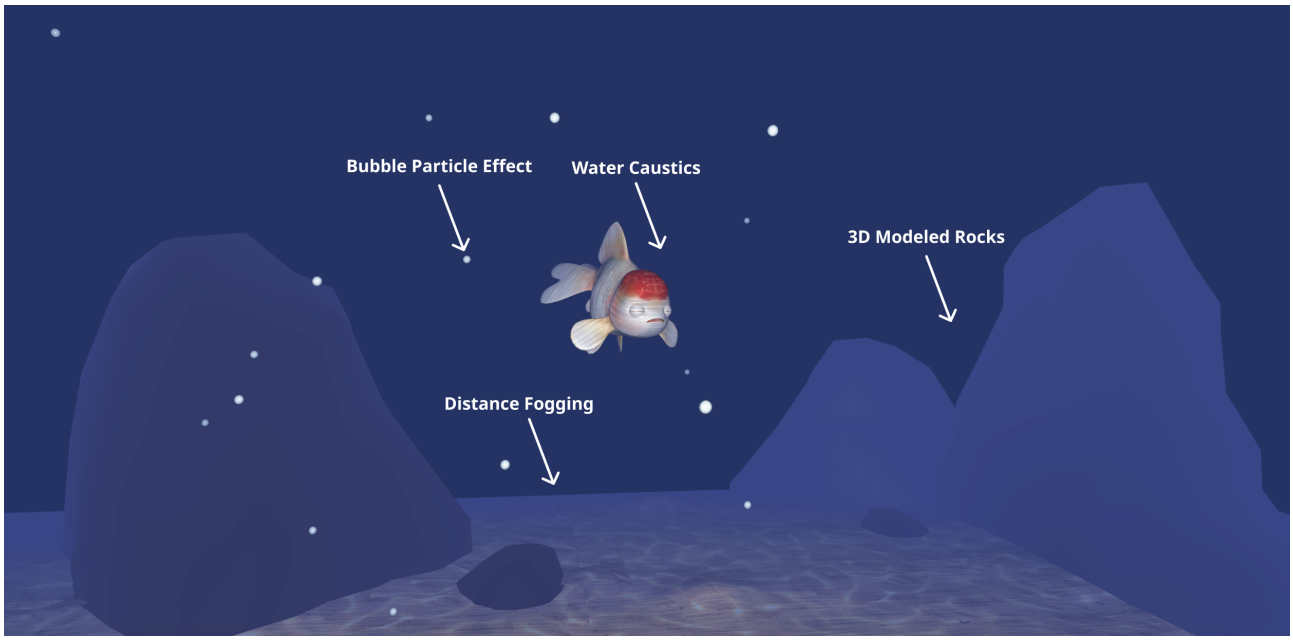

Fig. 4: Aquarium scene with bubble particle effect, water caustics, distance fogging, and 3D modeled rocks and ocean floor.

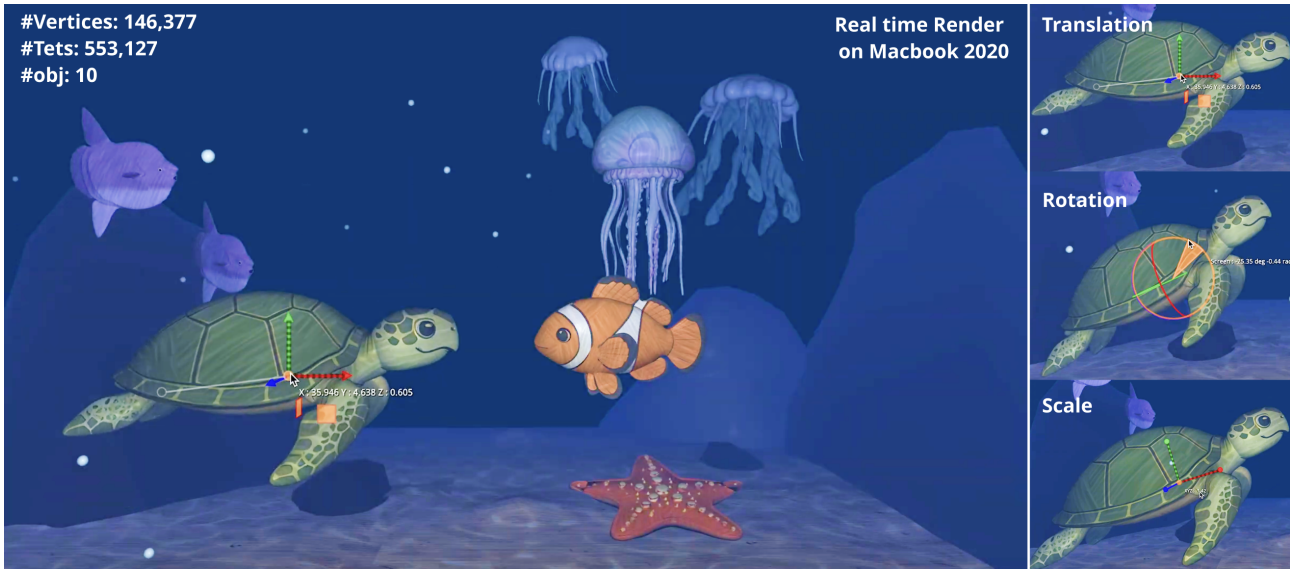

Fig. 5: Set up aquarium scene that supports secondary motion and translation, rotating, scaling for every fish. Rendered on Macbook 2020 Intel Chip computing 146,377 vertices and 553,127 tets in the scene total. You can switch affine handles and toggle each fish freely.

Key / Input	Action
`g`	Toggle Gizmo transform mode (translate / rotate / scale)
`s`	Save current fish positions to `fish_positions.json`
`l`	Load fish positions from `fish_positions.json`

5. Conclusion and Future Work

In this project, we implemented a complete pipeline that synthesized over ten high-quality fish models, remeshed them for physical simulation, and animated them using real-time physics-based deformation.

We addressed technical challenges such as gizmo handle implementation, secondary motion computation, and aquarium scene integration, including custom water shaders, lighting, and particle effects. This end-to-end system demonstrates the feasibility and complexity of building a fully interactive, physically-driven animation pipeline.

Looking ahead, we plan to extend the system with automatic rigging (e.g., Zhang et al. (2025)) and generate fish animation to further enhance scene dynamics. Additionally, we aim to implement material-aware mesh segmentation, using our introduced `secondary_motion_scale` parameter to modulate deformation across different regions. Finally, this pipeline could support physical fabrication by assigning 3D printing materials based on simulation parameters, enabling realistic, multi-material prints that mirror their real-life materials.

References

Google DeepMind. Introducing nano-banana pro. https://blog.google/technology/ai/nano-banana-pro/, 2025. Accessed: 2025-12-09.

Y. Hu, T. Schneider, B. Wang, D. Zorin, and D. Panozzo. Fast tetrahedral meshing in the wild. *ACM Trans. Graph.*, 39(4), July 2020. ISSN 0730-0301. doi: 10.1145/3386569.3392385. URL https://doi.org/10.1145/3386569.3392385.

Z. Lai, Y. Zhao, H. Liu, Z. Zhao, Q. Lin, H. Shi, X. Yang, M. Yang, S. Yang, Y. Feng, S. Zhang, X. Huang, D. Luo, F. Yang, F. Yang, L. Wang, S. Liu, Y. Tang, Y. Cai, Z. He, T. Liu, Y. Liu, J. Jiang, Linus, J. Huang, and C. Guo. Hunyuan3d 2.5: Towards high-fidelity 3d assets generation with ultimate details, 2025. URL https://arxiv.org/abs/2506.16504.

S. C. E. G. Y. Z. A. J. Otman Benchekroun, Jiayi Eris Zhang. Fast complementary dynamics via skinning eigenmodes. *ACM Transactions on Graphics*, 2023.

J. E. Zhang, S. Bang, D. I. W. Levin, and A. Jacobson. Complementary dynamics. *ACM Trans. Graph.*, 39(6), Nov. 2020. ISSN 0730-0301. doi: 10.1145/3414685.3417819. URL https://doi.org/10.1145/3414685.3417819.

J.-P. Zhang, C.-F. Pu, M.-H. Guo, Y.-P. Cao, and S.-M. Hu. One model to rig them all: Diverse skeleton rigging with unirig. *arXiv preprint arXiv:2504.12451*, 2025.

Acknowledgments

We would like to thank Professor Matusik and Professor Mina for their valuable guidance throughout the semester. TAs Selena Qiao and Raul Hernandez for their support and insight. Faraz Faruqi, Sean Zhan, and Clement Jambon for their feedback on our presentation. Special thanks to Yi Zhou for providing key insight and supporting us through the project.