A. Additional Experiments

We provide additional quantitative experiments and detailed discussion about Polygen [2] tokenization method here.

A.1. Experiments on Face Count Condition.

We tested the effectiveness of the face count condition by scaling the ground truth face count value by a scalar. We used the V2 model trained on the mesh dataset with fewer than 1600 faces as the test model and sampled 10K point clouds for each mesh to calculate the metrics. During inference, for each ground truth mesh, we input the scaled face count and the paired point cloud into the model and measured the effect on face count and mesh quality. As shown in Tab. 1, at a scale ratio of 0.8, the model significantly reduced the face count while maintaining mesh quality. However, at a scale ratio of 0.6, the face count did not decrease further, indicating that while the model has some ability to follow the face condition, it prioritizes mesh quality when the condition becomes difficult to meet. Similarly, when the scale ratio is set to greater than 1, the model exhibits similar behavior.

Table 1. **Experiments on Face Count Condition.** We control the face count condition using the scale ratio, where 1.0 indicates using the ground truth face count as the condition. The experiments show that our face count condition has the ability to control the number of faces.

Scale Ratio	CD↓ ($\times 10^{-2}$)	V_Ratio	F_Ratio
1.0	1.768	1.127	1.097
0.8	**1.734**	0.928	0.912
0.6	1.822	**0.902**	**0.882**
1.2	1.814	1.282	1.248
1.4	1.920	1.271	1.252

A.2. User Study.

We conducted a user study on mesh generation quality to compare MeshAnything [1] and our works. For a fair comparison, we randomly sampled 30 meshes with fewer than 800 faces from the evaluation dataset and input their paired point clouds into each model. Users were asked to select the result they preferred from the two options. We collected responses from a total of 43 users, and the voting rates for [1] and MeshAnything V2 were 32.2% and 67.8%, respectively. This indicates that the results generated by V2 are more aligned with human preference.

A.3. Discussion on PolyGen Tokenization Method.

Polygen[2] introduces a mesh tokenization approach that differs from other existing methods [3] for mesh generation. It first employs an autoregressive vertex model to generate the 3D coordinates of the mesh's vertices. These vertices are then used as a prefix and fed into another autoregressive face model, which connects these vertices into faces, thus constructing the entire mesh. Since the 3D coordinates are already provided by the vertex model, the face model does not need to estimate the 3D coordinates, but only specifies the connections using vertex indices, significantly reducing the sequence length. During the face generation, this tokenization method can be combined with AMT further shorten the token sequence length.

By using vertex indices to represent faces, PolyGen [2] tokenization consumes only one token to define a face, whereas other methods require three tokens to represent a single vertex. Assuming a vertex is referenced n times, PolyGen's tokenization requires $3 + n$ tokens, whereas other tokenization methods would require $3 \times n$ tokens. Although PolyGen spends one additional token when a vertex is referenced only once, in most cases, each vertex is referenced multiple times, allowing PolyGen's approach to save sequence length.

Although PolyGen tokenization effectively reduces the token sequence length, this generation method requires the model to accurately predict vertex positions first, which may not be ideal for an autoregressive model.

In the experiments section of our main paper, we compare Polygen tokenization method with other methods. To make a fair comparison, we merge the two stage generation process of PolyGen [2] into a single model that first generates vertex coordinates and then expresses the faces by generating vertex indices. Besides, we combine PolyGen's face generation stage with AMT to further reduce the token sequence length.

B. Limitations.

Although there is a large improvement over V1, the accuracy of MeshAnything V2 is still insufficient for industrial applications. More efforts are needed to improve the model's stability and accuracy.

References

[1] Yiwen Chen, Tong He, Di Huang, Weicai Ye, Sijin Chen, Jiaxiang Tang, Xin Chen, Zhongang Cai, Lei Yang, Gang Yu, Guosheng Lin, and Chi Zhang. Meshanything: Artist-created mesh generation with autoregressive transformers, 2024. 1

[2] Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *International conference on machine learning*, pages 7220–7229. PMLR, 2020. 1

[3] Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. Meshgpt: Generating triangle meshes with decoder-only transformers. *arXiv preprint arXiv:2311.15475*, 2023. 1