Pacific Graphics 2022
N. Umetani, E. Vouga, and C. Wojtan
(Guest Editors)

Volume 40 (2021), Number 7

MoMaS: Mold Manifold Simulation for real-time procedural texturing Supplementary Materials

F. Maggioli[1], R. Marin[1], S. Melzi[2] and E. Rodolà[1]

[1]Sapienza - University of Rome, Italy
[2]Università di Milano Bicocca, Italy

In this document we provide additional information and more detailed insights on the experimental results presented in the main paper. Moreover, we give additional details about our implementation of the algorithm for moving a particle over a surface.

1. Particle Motion Over a Surface

Algorithm 1 summarizes the entire process. The procedure TANGENTSUBSTEP is given a point p in texture space and the triangle t where p lies. It computes the movement over the tangent space of the mesh along some vector Δp. The vector is still defined in texture space, and thus must be rescaled according to the local metric tensor $\mathbf{g}(t)$ (Line 3). The final position after the movement is converted to barycentric coordinates to determine if the border of the triangle has been crossed (Lines 5-10). The adjacent triangle w closest to the final point is selected and the point is projected on it (Lines 12 and 13). Finally, the coordinates are converted to 2D via barycentric coordinates (Lines 15 and 16). To handle meshes with boundaries, it is sufficient to identify if the agent crossed a boundary edge (*i.e.* there is no adjacent triangle on that edge). If this is the case, we make the agent bounce on the edge.

The vectors n_τ and c_τ are, respectively, the normal and the barycenter of a triangle τ, while $\mathrm{Adj}(\tau)$ is the set of its adjacent triangles. The matrices $\mathbf{T}_\tau$ and $\mathbf{L}_\tau$ are defined for each triangle τ as

$$\begin{aligned} \mathbf{T}_\tau &= \begin{pmatrix} r_{\tau,1} - r_{\tau,3} & r_{\tau,2} - r_{\tau,3} \end{pmatrix} \in \mathbb{R}^{2\times 2} \\ \mathbf{L}_\tau &= \begin{pmatrix} v_{\tau,1} & v_{\tau,2} & v_{\tau,3} \end{pmatrix} \in \mathbb{R}^{3\times 3} \end{aligned} \tag{1}$$

where $r_{\tau,i}$ are the coordinates of the vertices of τ in texture space and $v_{\tau,i}$ are the coordinates of the vertices of τ in 3D space. The matrix $\mathbf{T}_\tau$ is the conversion matrix from barycentric to edge coordinates and is such that $\mathbf{T}_\tau \lambda' = p - r_{\tau,3}$, given that λ' are the first two components of the barycentric coordinates of p. Matrix $\mathbf{L}_\tau$ is the conversion matrix from barycentric to Cartesian coordinates. Usually, it is defined as a rectangular matrix in $\mathbb{R}^{4\times 3}$, since it must have a row of ones in order to ensure the point is inside the triangle. In our case, we need to allow points to move outside triangles, and thus we remove that row. As a side effect, we can move back and forth between Cartesian and barycentric coordinates by inverting $\mathbf{L}_\tau$, which saves much computation with respect to the classic area-based method.

The procedure TANGENTSTEP iterates the process multiple times to continuously determine the new position and direction in texture space, as well as the current triangle. The number of substeps N is some fixed constant depending on the mesh resolution and it is used to determine the length of the sub-step (Line 3). At each iteration, the algorithm moves the agent by a small step (Line 6) and then uses a slightly larger step to determine the next direction (Line 8). The direction is then normalized and rescaled to a proper length (Line 9). The call to TANGENTSUBSTEP also has the job of determining the next direction and the triangle of the next position to pass them to the next iteration.

2. Area Coverage and Parameters Tuning

Figure 1 accompanies Figure 8 from the main paper. Differently from the evaporation rate, the movement speed of the agents and their angle of vision seems to not affect the area coverage. Even if changing these parameters produces very different patterns, the portion of surface area covered by the pheromone trace does not vary. The plots for the agents' turn speed and the sensors' distance behave similarly.

Figure 10 in the main paper shows an interpolation between simulation parameters, in order to give an insight on how a single parameter affects the behavior of the resulting pattern. Figure 2 and Figure 3 show a more detailed overview of the parameters' interpolations. Figure 2 accompanies Figure 10 from the main paper, whereas Figure 3 analyzes the effect of changing the vision distance and angle parameters. In both cases, the parameters are linearly interpolated using the ranges in Table 1 from the main manuscript. The turning speed of the agents does not seem to affect the pattern over the surface. Instead, we observed a correlation between turning speed and how fast the pattern changes during its evolution. Moreover, we notice that changing the movement speed and the evaporation rate has a higher visual impact than changing the sensors' parameters. A small change in the latter is visually more subtle.

Algorithm 1 Step in the tangent space of a mesh.

```
1: procedure TangentSubStep($p$, $\Delta p$, $t$)
2:     // Rescale the displacement vector
3:     $\Delta p \leftarrow \frac{\Delta p}{\sqrt{(\Delta p)^\top \mathbf{g}(t) \Delta p}}$
4:     // Convert to barycentric coordinates
5:     $q \leftarrow p + \Delta p$
6:     $\lambda \leftarrow \mathbf{T}_t^{-1}(q - r_{t,3})$
7:     $\lambda_3 \leftarrow 1 - \lambda_1 - \lambda_2$
8:     if $\lambda$ is inside $t$ then
9:         return $(t,\ p + \Delta p)$
10:    end if
11:    // Search for the nearest adjacent triangle
12:    $w \leftarrow \arg\min_{\tau \in \mathrm{Adj}(t)} \{|\langle n_\tau,\ q - c_\tau \rangle|\}$
13:    $q \leftarrow q - \langle n_w,\ q - c_w \rangle n_w$
14:    // Return final texture coordinates and the new triangle
15:    $\lambda \leftarrow \mathbf{L}_w^{-1} q$
16:    return $\left(w,\ \lambda_1 r_{w,1} + \lambda_2 r_{w,2} + \lambda_3 r_{w,3}\right)$
17: end procedure
1: procedure TangentStep($p$, $\Delta p$, $t$)
2:     $p' \leftarrow p$
3:     $\Delta p' \leftarrow \frac{\Delta p}{N}$
4:     for $i \leftarrow 1$ to $N$ do
5:         // Compute the final position
6:         $\left(w,\ p''\right) \leftarrow$ TangentSubStep($p'$, $\Delta p'$, $t$)
7:         // Compute the new direction
8:         $\left(-,\ \Delta p''\right) \leftarrow$ TangentSubStep($p'$, $\Delta p'$, $t$)
9:         $\Delta p' \leftarrow \|\Delta p'\| \frac{\Delta p'' - p''}{\|\Delta p'' - p''\|}$
10:        $p' \leftarrow p''$
11:        $t \leftarrow w$
12:    end for
13:    return $\left(t,\ p',\ \mathrm{atan2}\left(\Delta p'\right)\right)$
14: end procedure
```

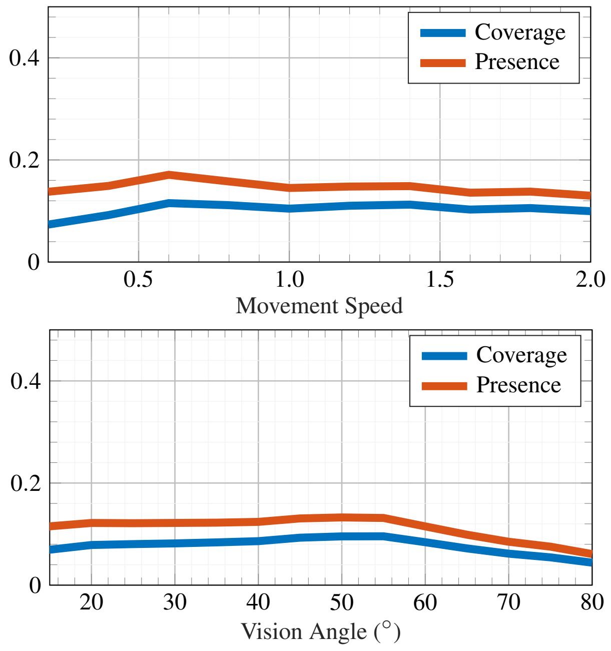

Figure 1: *Coverage and presence at varying of the movement speed and the vision angle. It is evident that the two parameters does not affect significantly the area covered by the pheromone.*

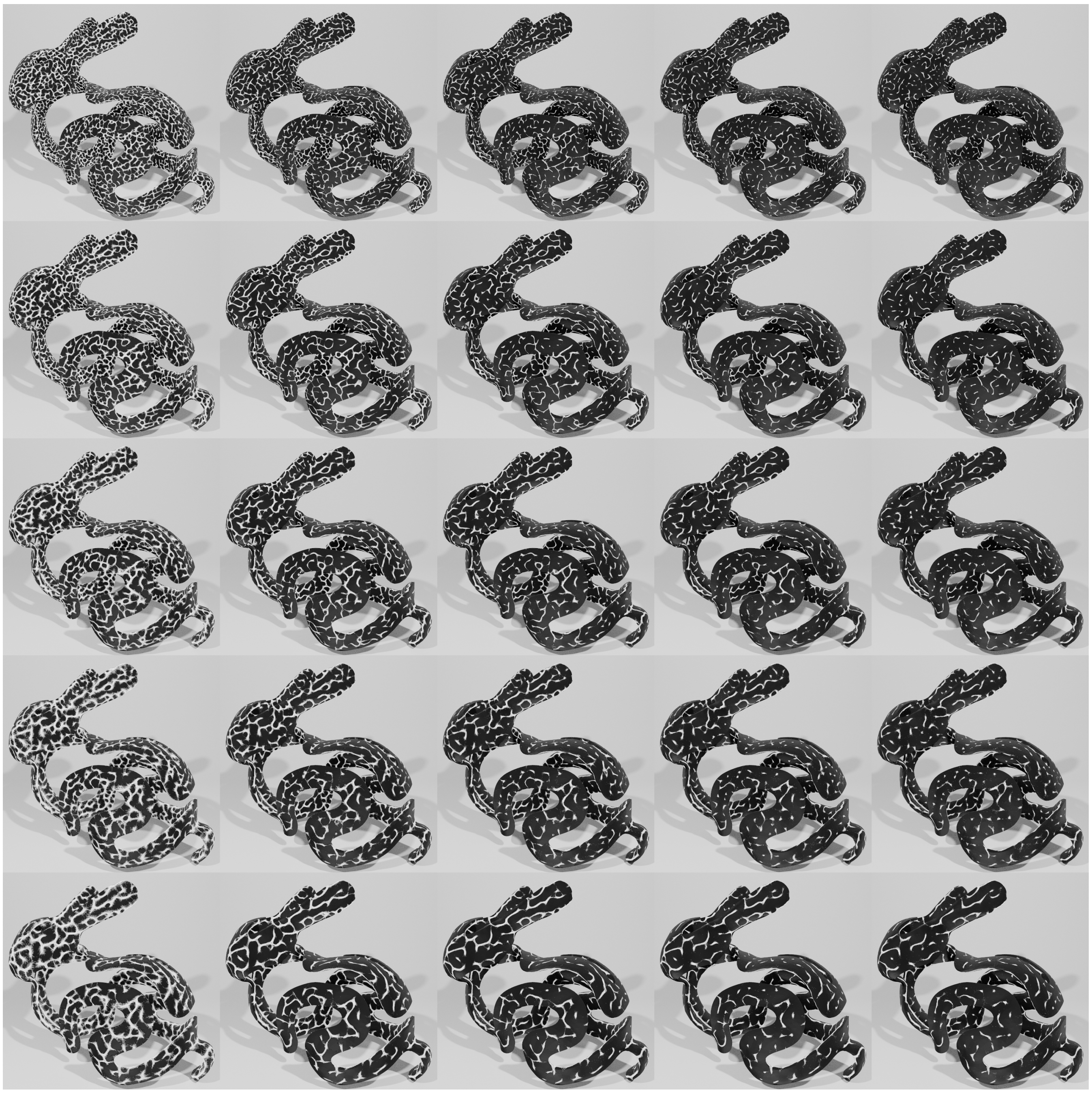

Figure 2: *More samples for the patterns resulting from the interpolation of the simulation parameters regulating the agents' movement speed and the pheromone's evaporation rate. Refer to Figure 10 from the main article.*

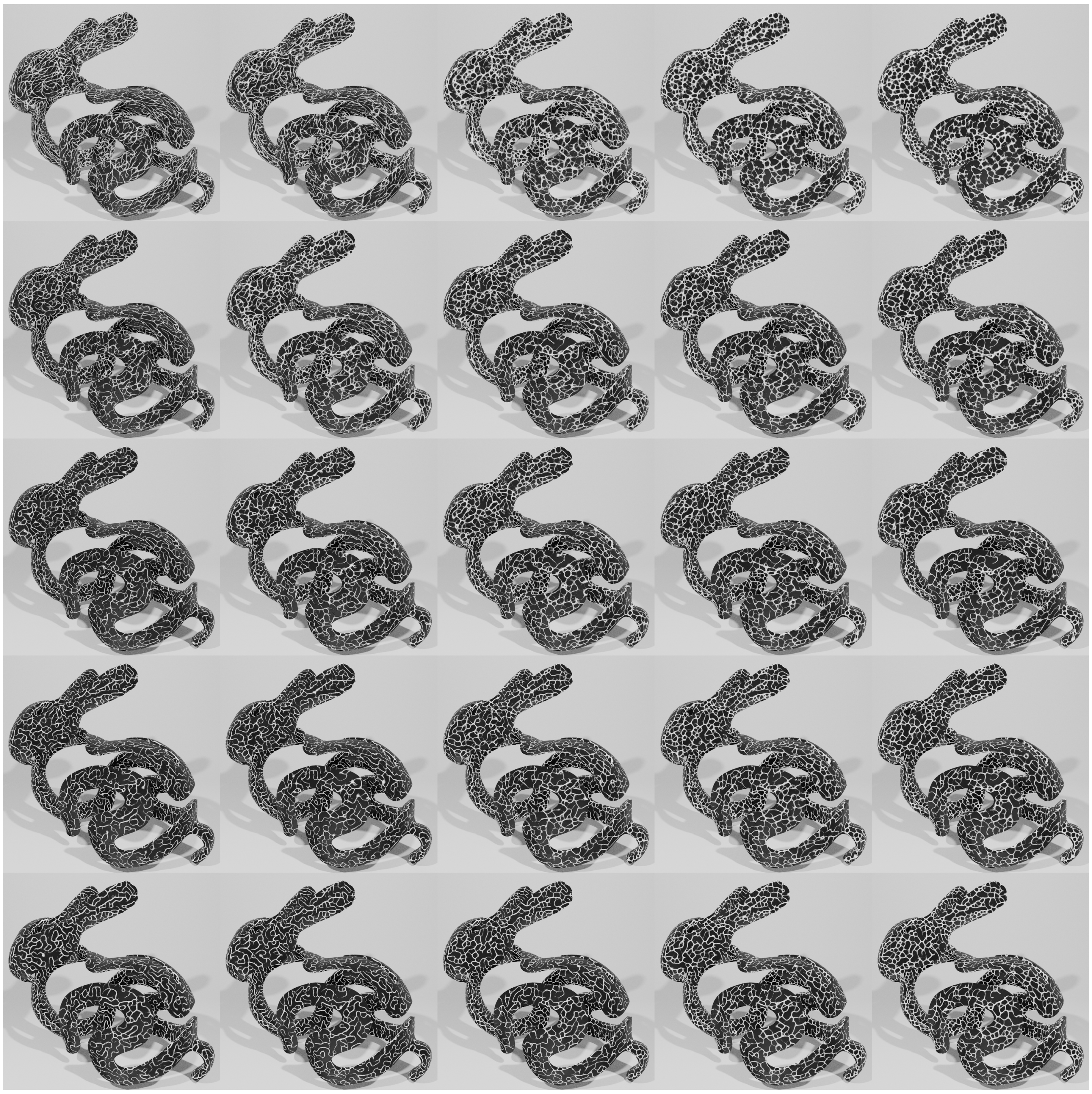

Figure 3: *More samples for the patterns resulting from the interpolation of the simulation parameters regulating the sensor angle and distance.*

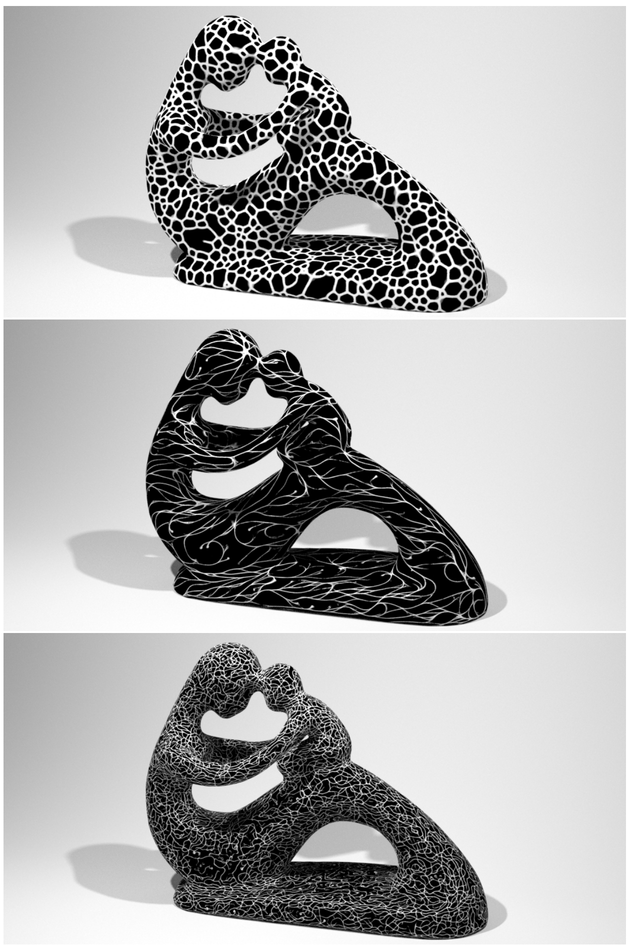

Figure 4: *The textures used as a base for the materials shown in Figure 11 of the main article.*

3. Patterns for materials

Figure 11 in the main paper shows three examples of different patterns that can arise from the slime mold algorithm on surfaces by tuning the simulation parameters. The materials used for the examples are fairly complex, and they come from the mixing of different shaders. However, the visible patterns are the result of a slime mold evolution, and we show them in Figure 4. The top texture has been used as a filter, for mixing two shaders and produce lava rivers on a rock, and as a displacement map for enhancing the details. We did the same for the bottom texture when producing the moss material over the bark. The central texture has been used as an inverse bump map on a metallic material to produce scratches and engravings.

Tables 2, 3 and 4 from the main paper summarize quantitative results on the algorithm's performance. The results are averaged over a sample of triangle meshes with various resolutions. We show the sample of 10 meshes used for these experiments in Figure 5 and we summarize their statistics in Table 1.

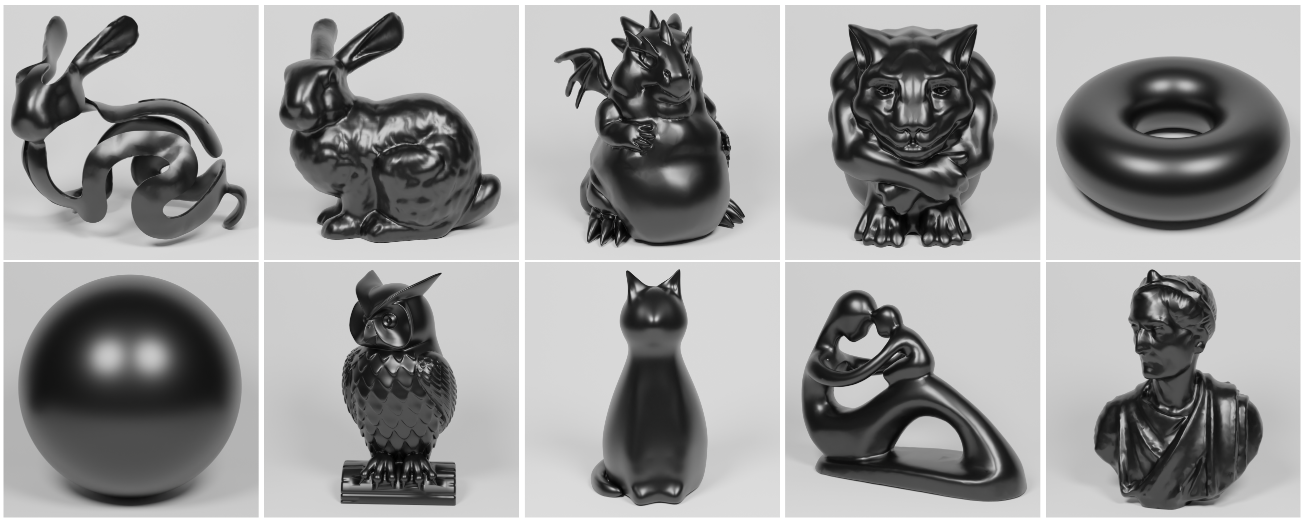

Figure 5: *The set of meshes used for testing performance.*

mesh	num. verts	num. edges	num. tris
bunny stripes	18k	55k	37k
bunny	35k	104k	69k
dragon	32k	95k	63k
gargoyle	20k	59k	39k
torus	2k	7k	5k
sphere	32k	98k	65k
owl	39k	118k	79k
cat	64k	193k	129k
fertility	10k	30k	20k
roman bust	14k	42k	28k

Table 1: *Statistics of the sample meshes shown in Figure 5.*