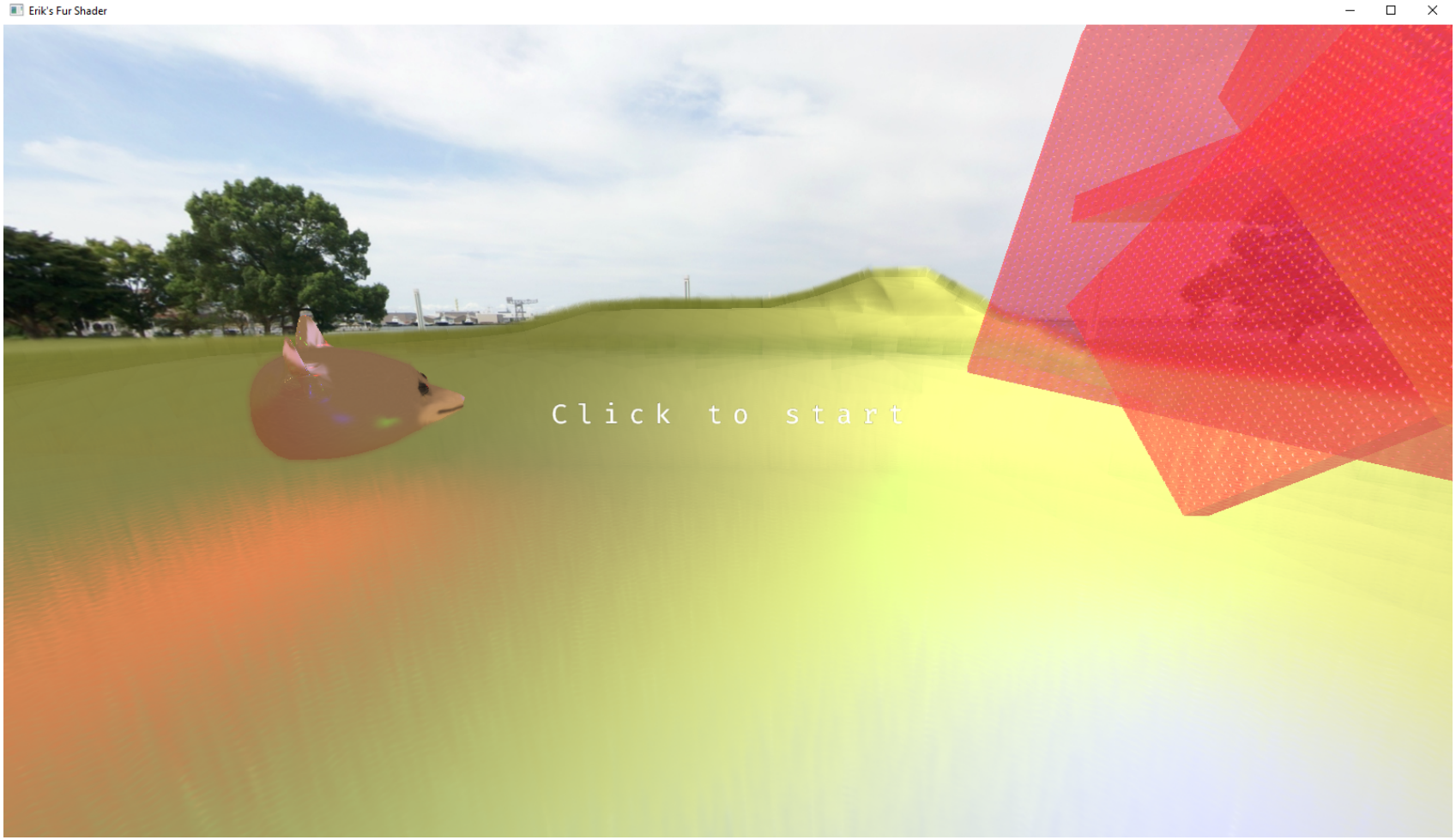
Fur rendering with shells, fins, and order-independent transparency

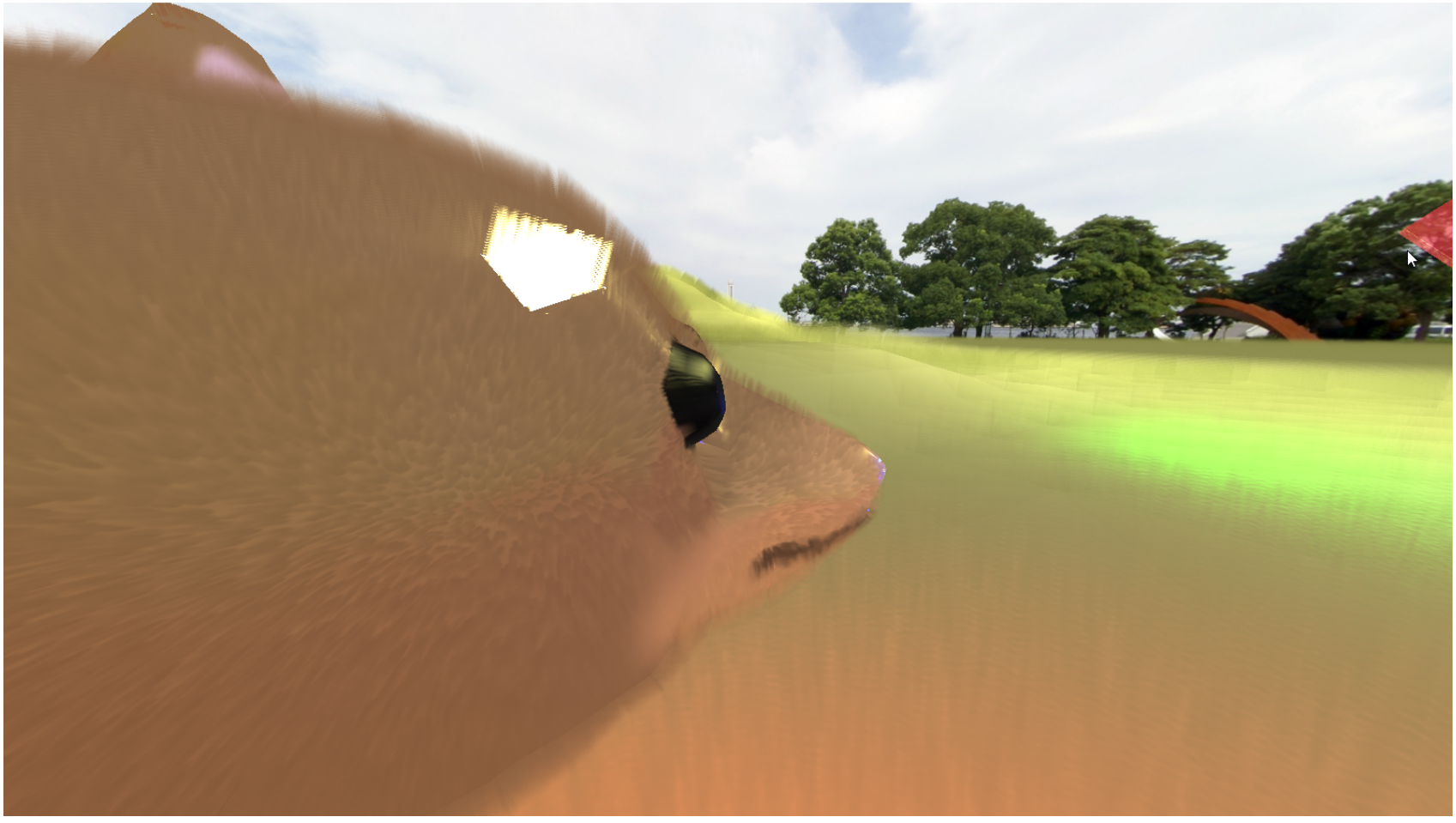
Odd-Erik Frantzen
https://github.com/odderikf/fur_project

April 19, 2023

Results:



Results:



Elements of the scene:

- Things from previous exercises
  - Phong lighting with dynamic pointlight sources
  - Textured, normal mapped, roughness mapped objects.
- Skybox at infinity
- Blender-exported object files and painted textures
- Shell-method fur volume generation
- Fin-method fur-card generation for better silhouettes
- Blending shader for semitransparent elements

Skybox

Yokohama2 by Emil "Humus" Persson, CC-BY 3.0

Skybox

Cubemap texture (6 images, samplerCube) mapped onto a box around origin with radius 1

Skybox

TF:

```
glm::mat4 TF = VP; // no model
mvp = glm::translate(TF, -cameraPosition);
```

Vertex shader:

```
gl_Position = TF * vec4(position, 1.0f);
gl_Position = gl_Position.xyww;
uv_out = origin-position;
```

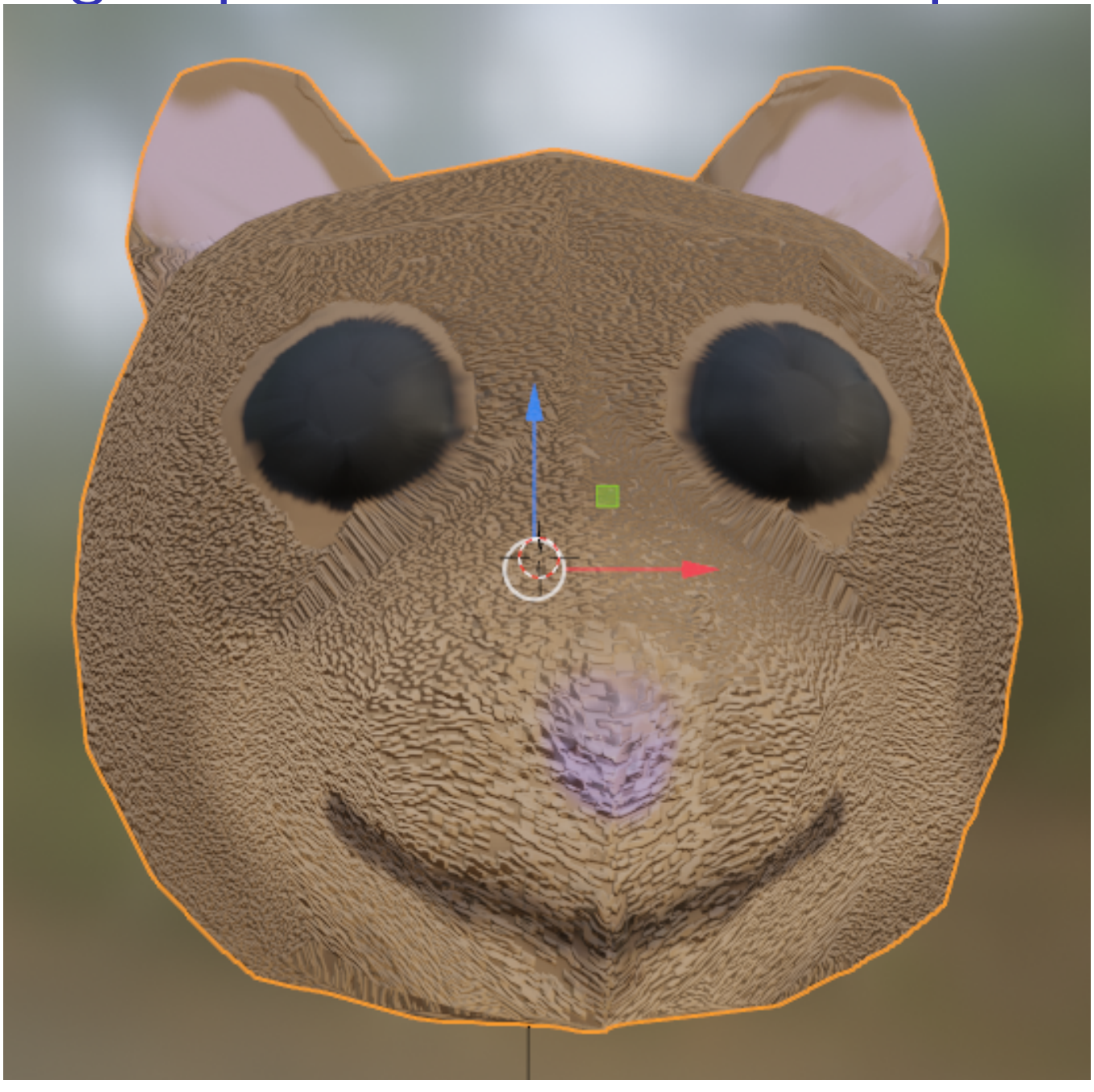
Fragment Shader:

```
color.rgba = texture(skybox, uv);
```

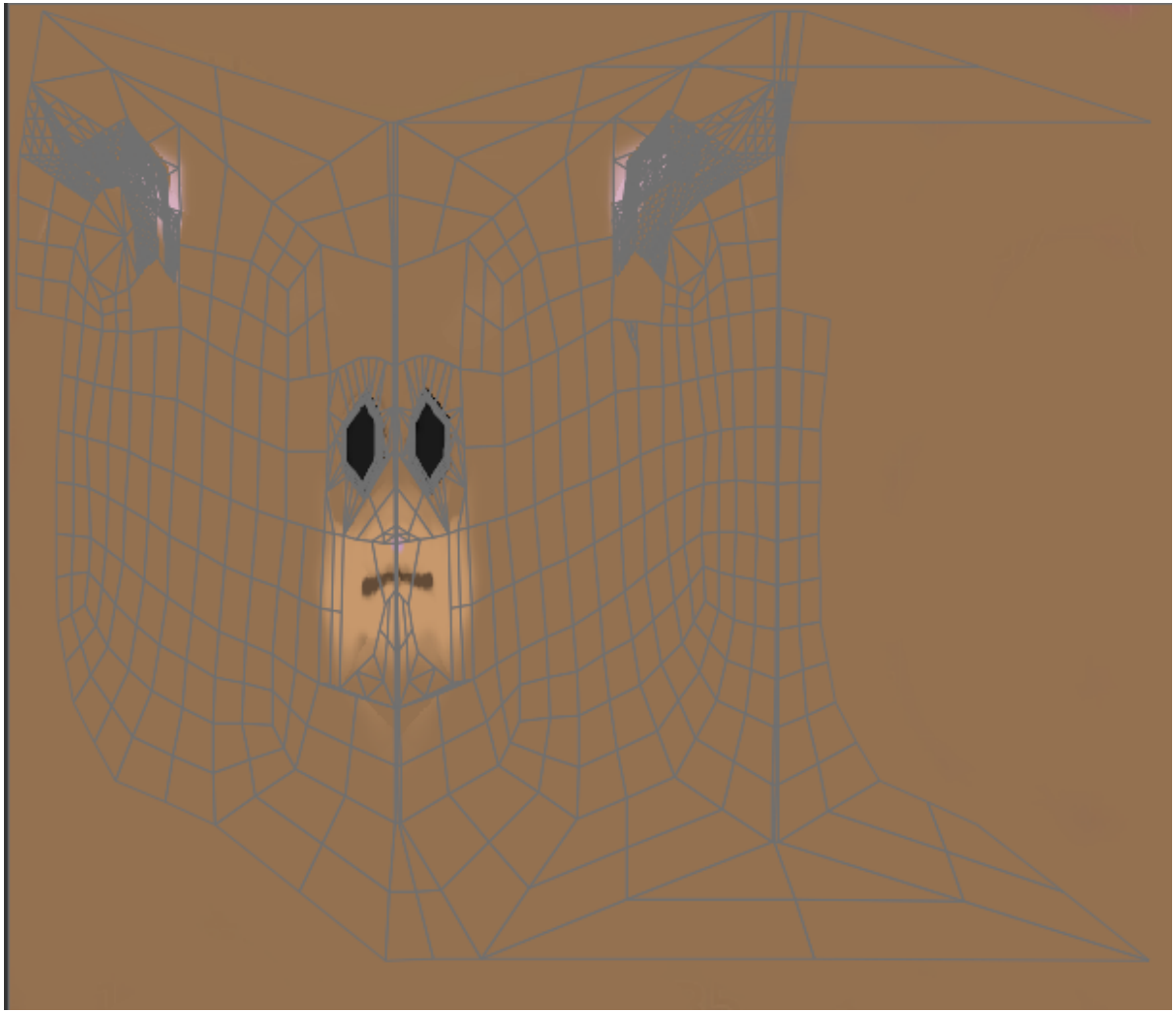
Blender objects

Disclaimer: I am not good at blender.
Models generated through normal modeling in blender.
Basic shading set up to aid intuition
UV-generation
Texture-painting directly on model
Baked normals

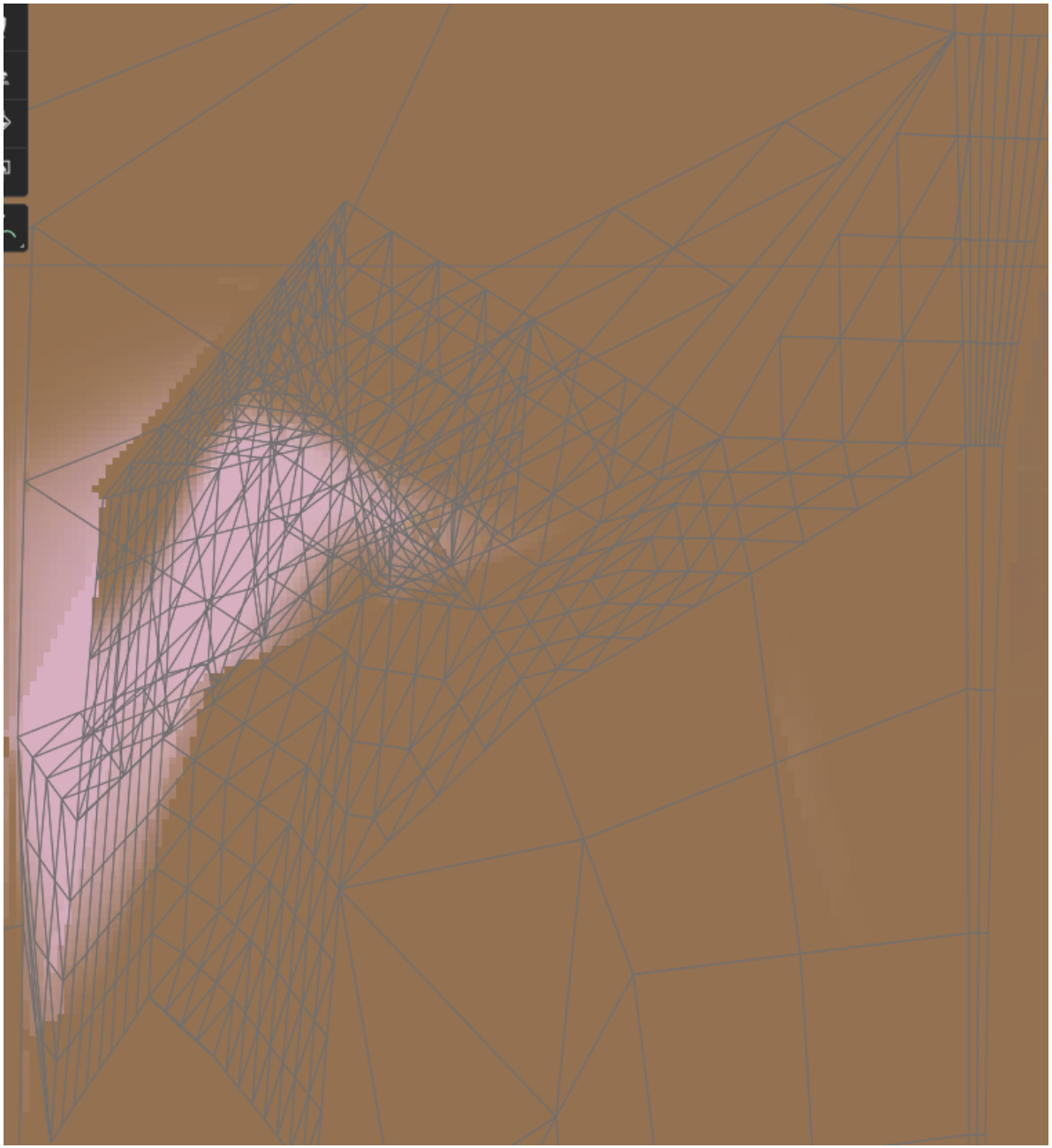
Trying to preview fur ...Wasn't super helpful



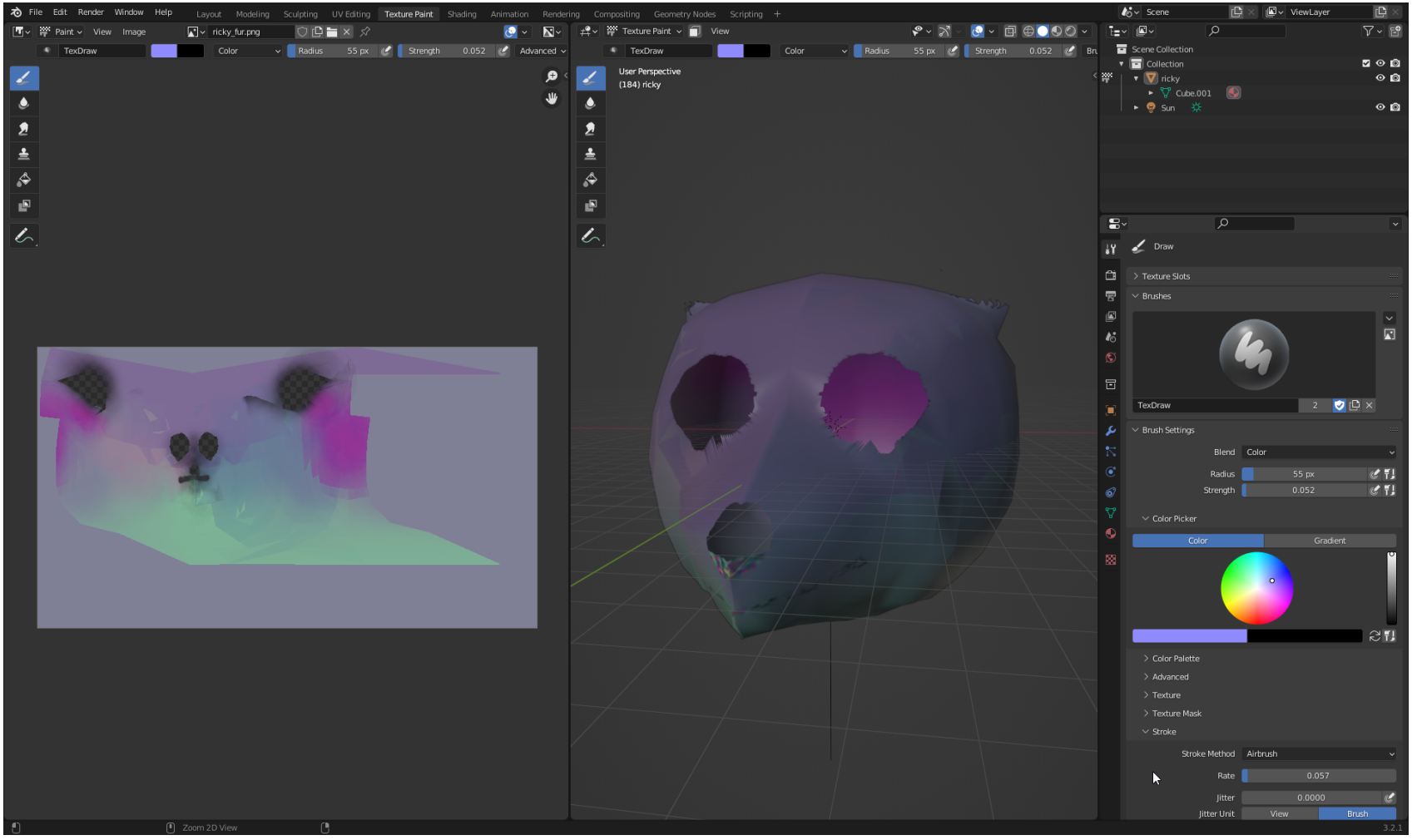
UVs are annoying, huh?



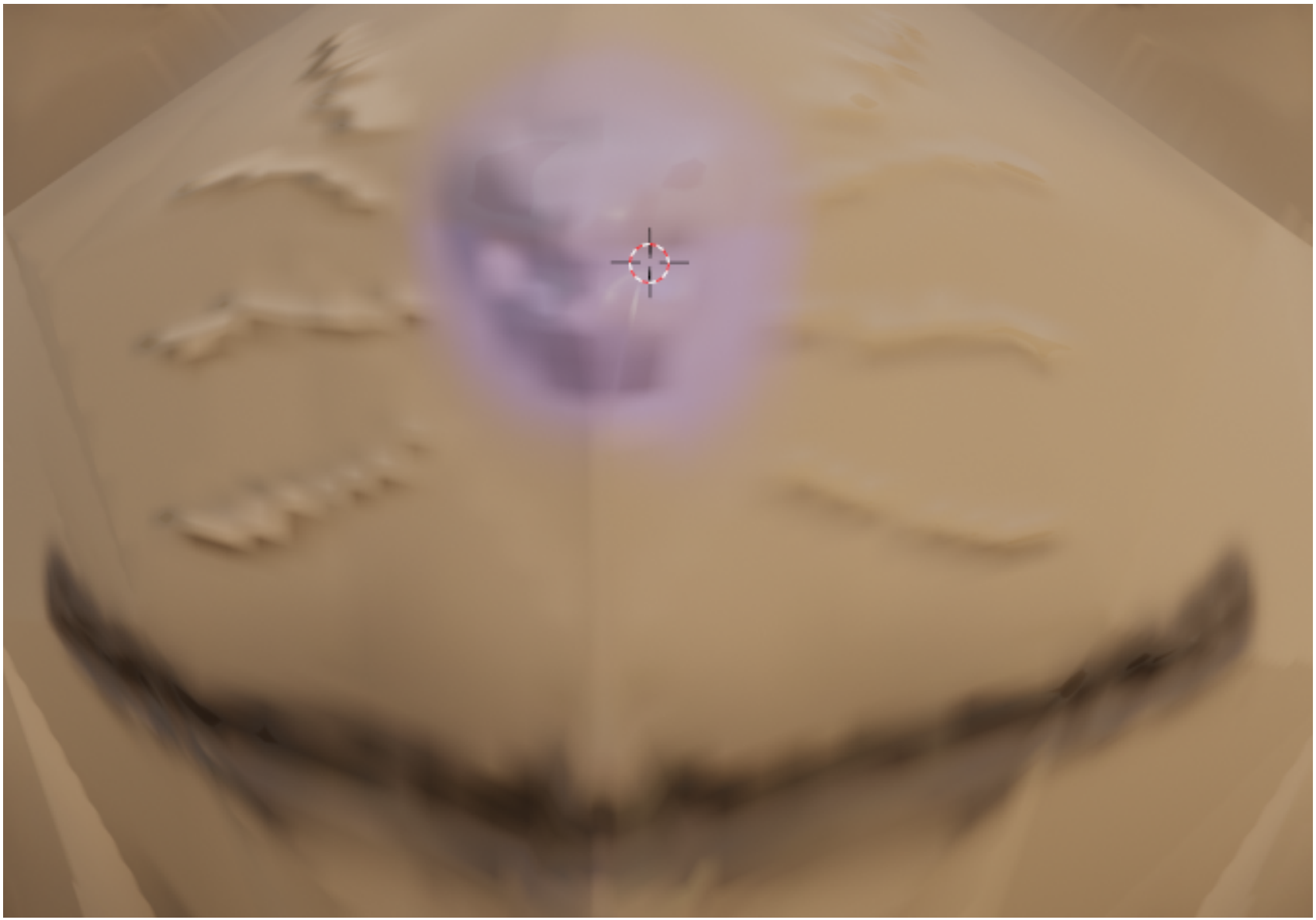
...I messed up the ear area



Texture painting



displacement map baked normals

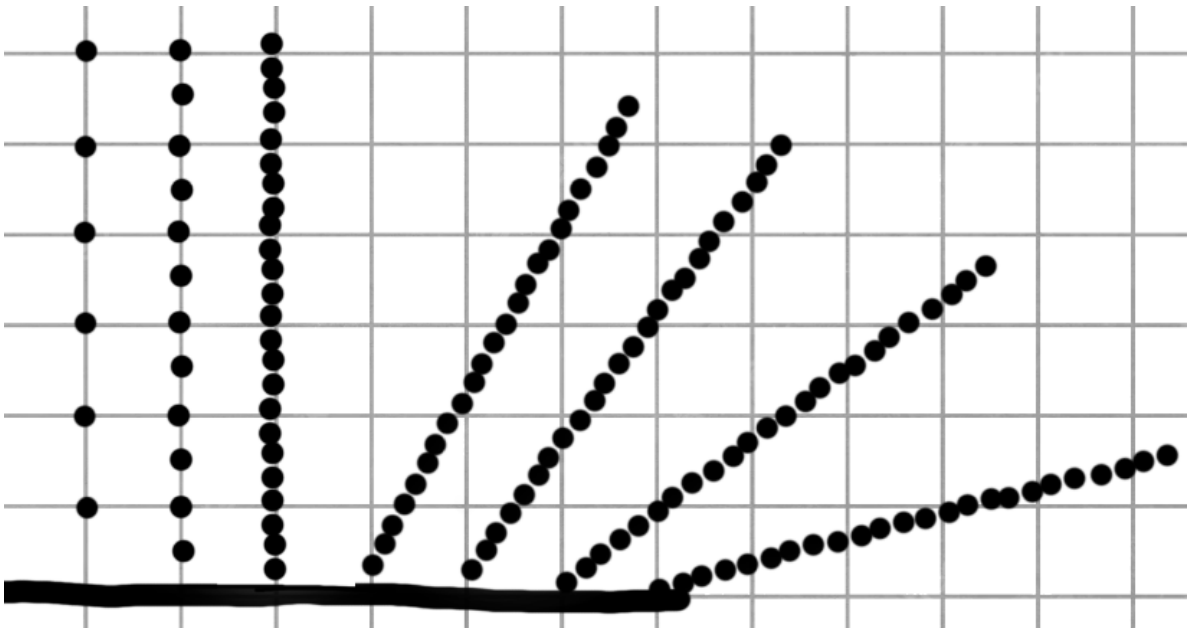


Shell method fur generation

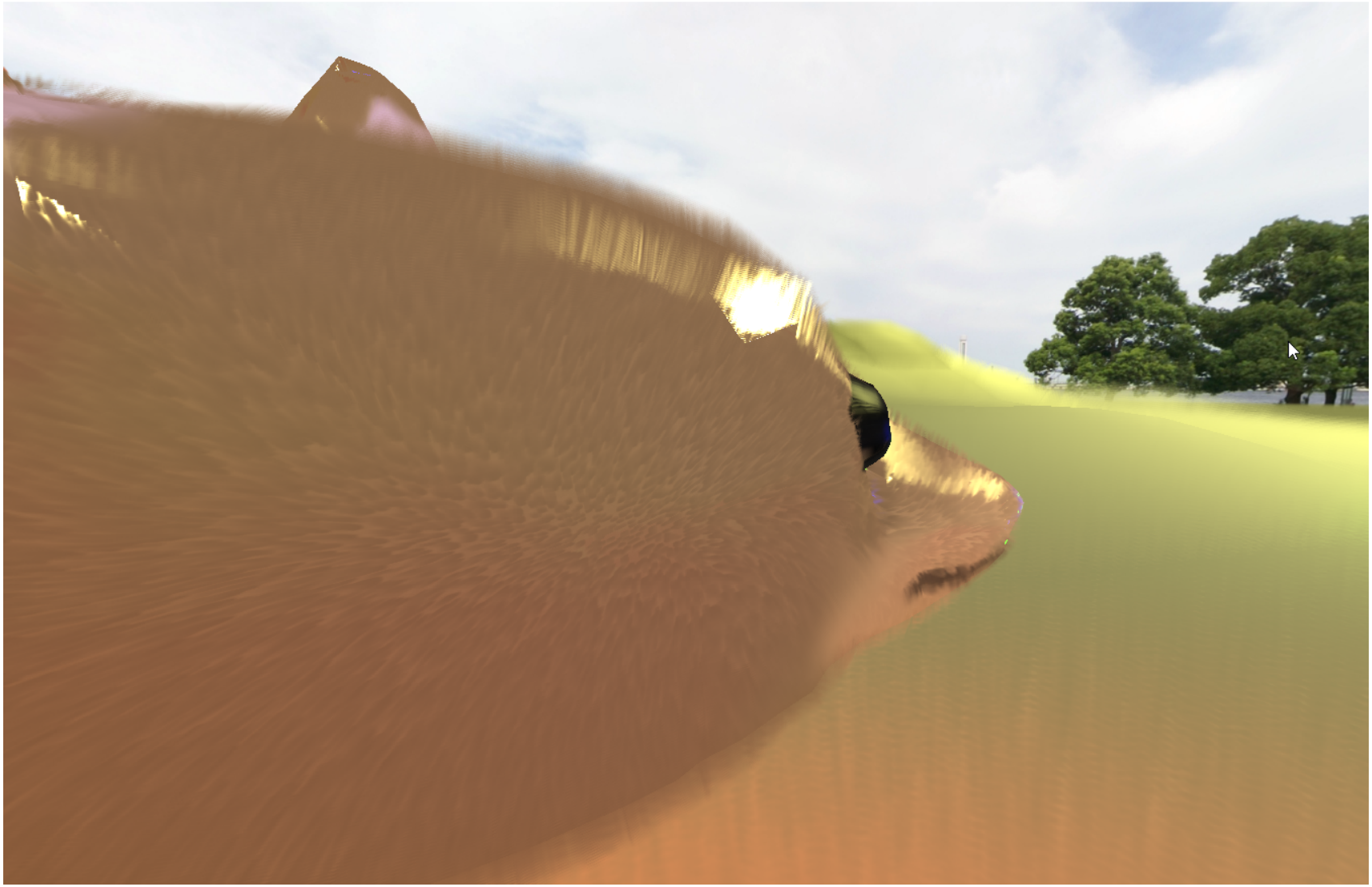
Render solid base
Render shells with geometry shader
Triangle $\rightarrow$ several displaced copies
Texture lookup determines direction and length of displacement
A uniform also scales displacement
Can add uniform vector for further dynamic displacement (wind, interaction)

Forming Strands

Render mostly transparent shells, with 'dots' of visibility.
These dots form lines, strands of fur.
Darker roots, brighter tips, for cheap 'self-shadowing'
'Thinning' through lowering alpha along length.
Could use 3D textures



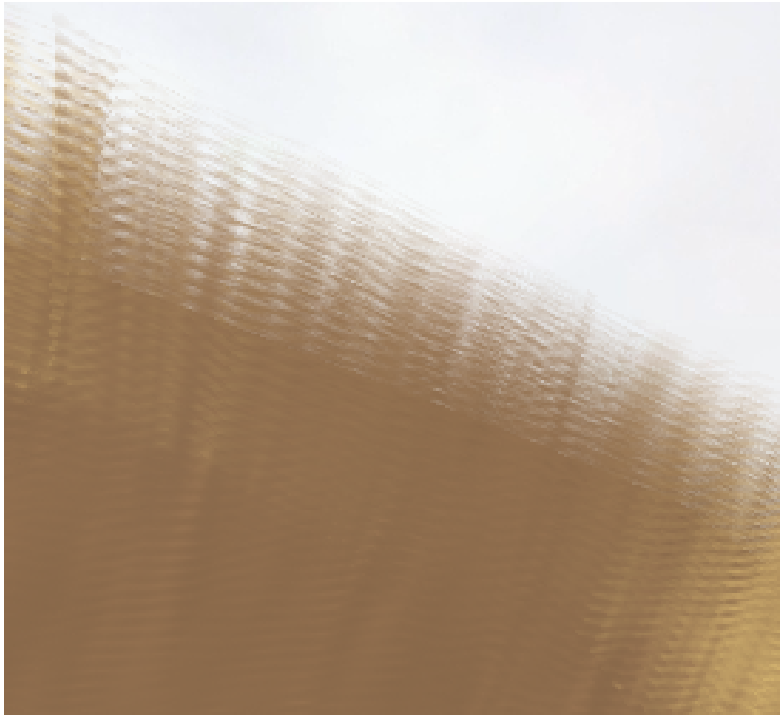
Shell result:



Dots texture:

Dots are stored as a texture,
generated with a simple python script.
Could replace this with 3d texture,
Could scale UVs (density uniform?)
to loop faster than other textures to save on space
Could make it implicit with some function in shaders
Could rig it to be more localized

Downside



Fin method fur generation

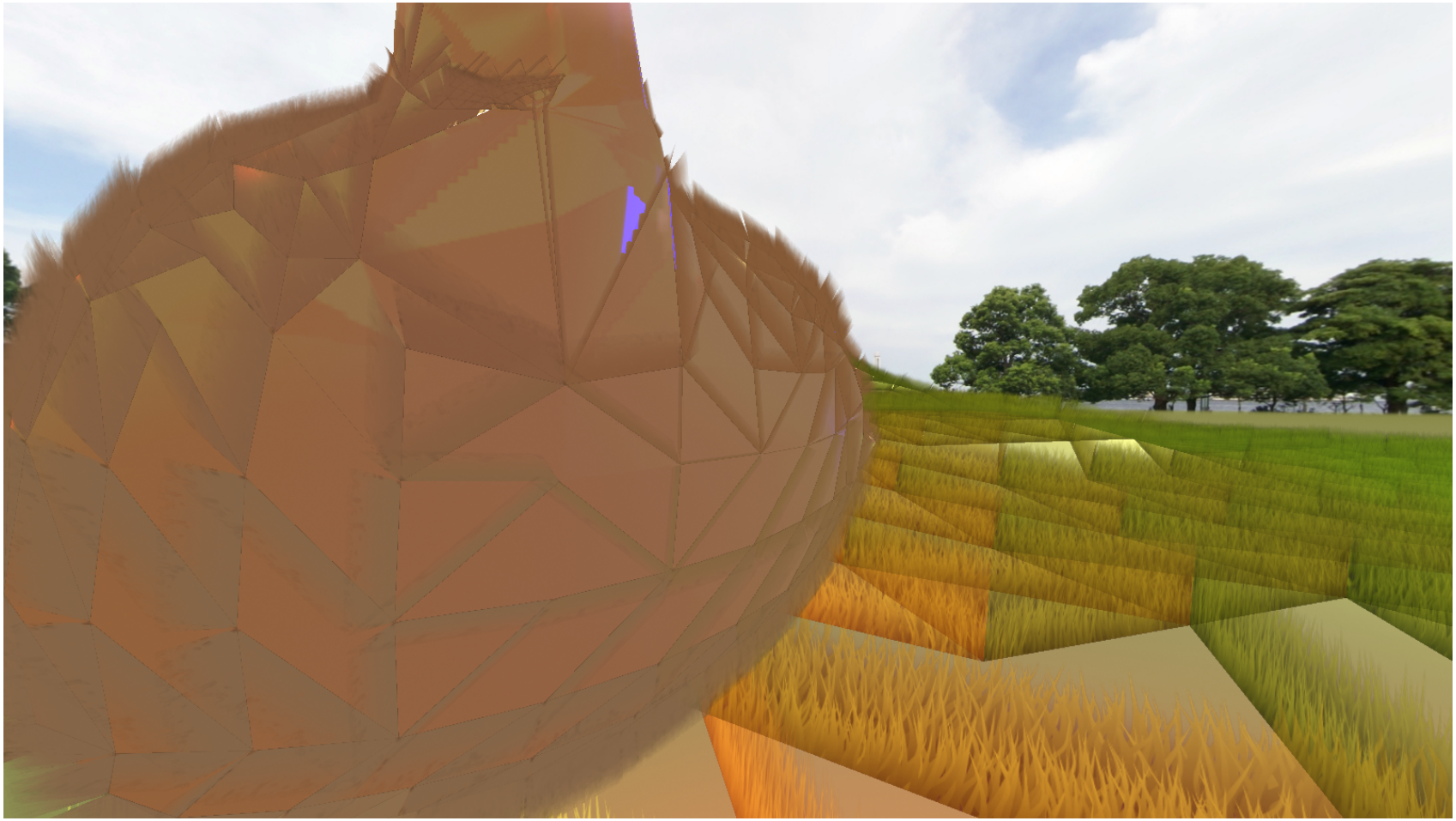
Dynamically generate fur cards
(Billboards that show fur strands)
Helps cover the silhouette,
where you see between the layers of the shells better
This is also done with a geometry shader

Fin method fur generation

Idea: edge $\rightarrow$ billboard
NVIDIA: adjacent triangles $\rightarrow$ billboard
My solution: triangle $\rightarrow$ 3 billboards (as triangle-strips)
Only edges at/near silhouette
Fade in and out
Draw if camera-to-point is near-normal on edge-normals.
Billboards follow fur direction away from original edge
Density determined by edge density
Segmented vertically to enable 3d effects (like bending nonlinearly in wind)

Fin Method results

Fins do not easily sell volume

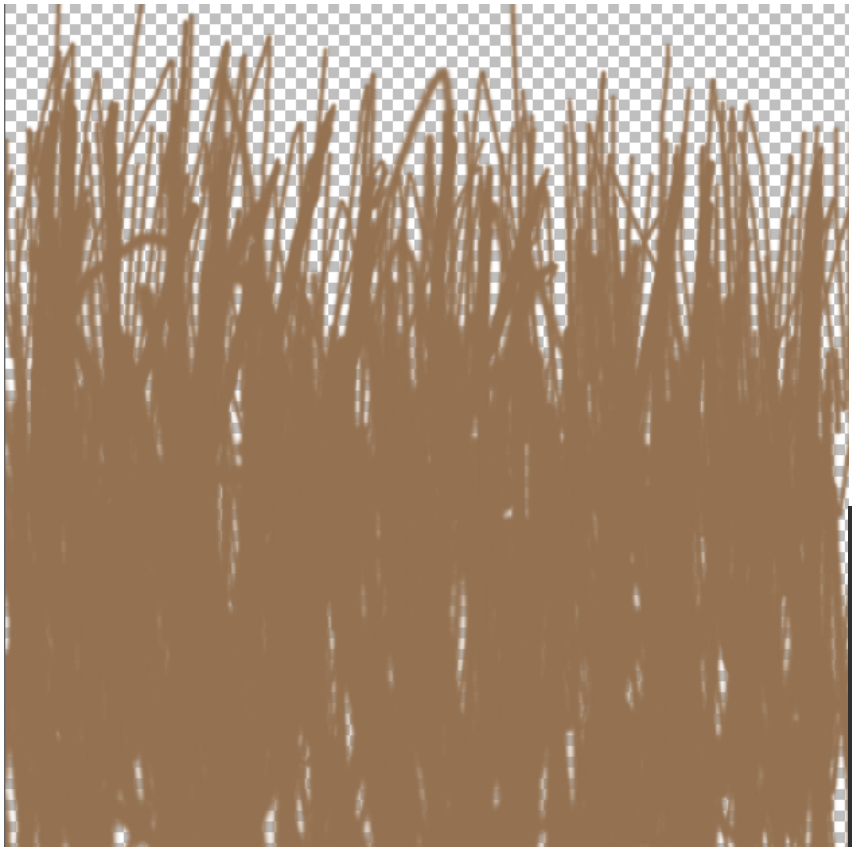


Fin Method results

They do work great at the silhouette though

Fins textures

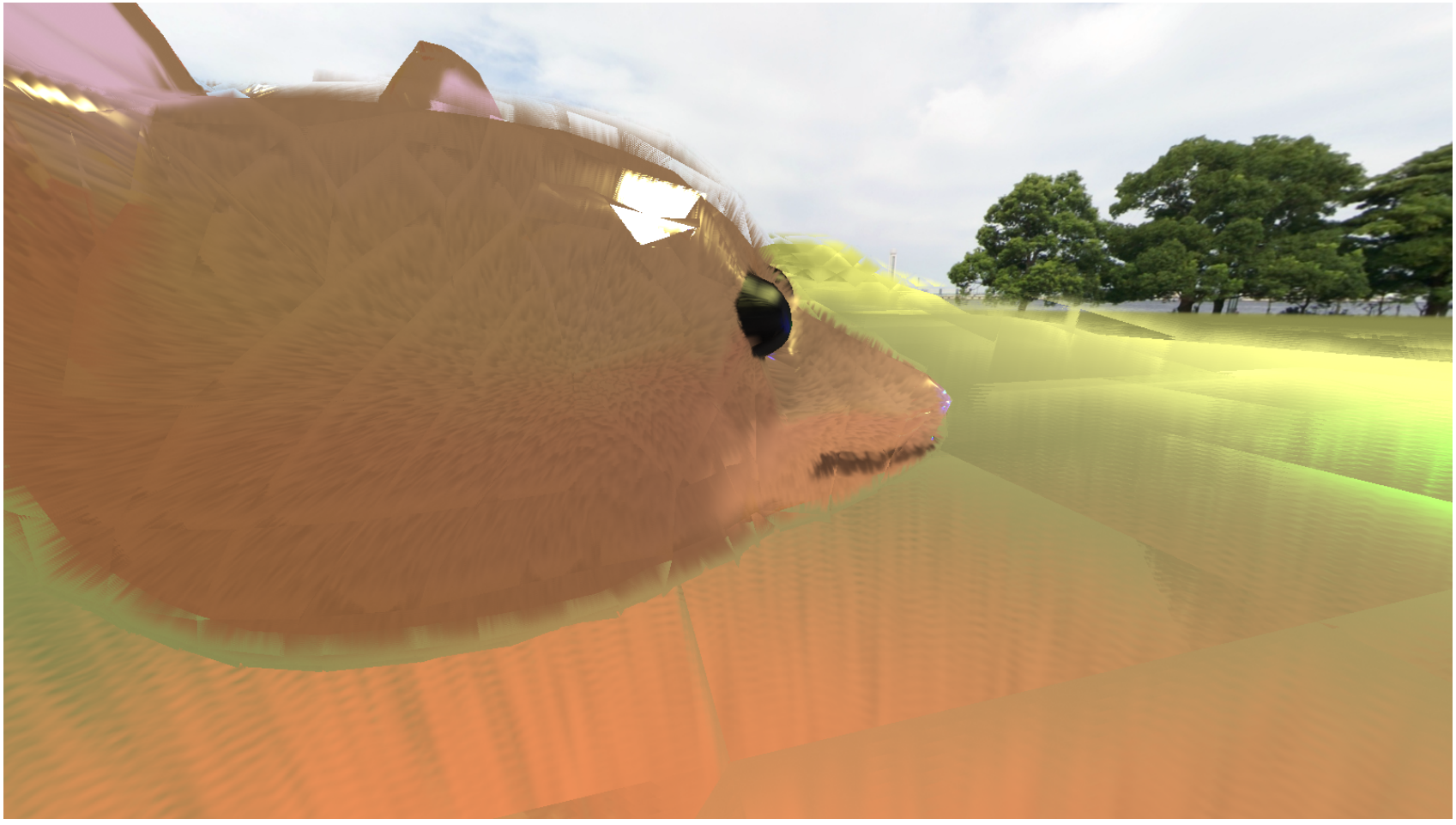
The grass texture was found free online https://pixabay.com/vectors/grass-background-border-green-grass-2029768/



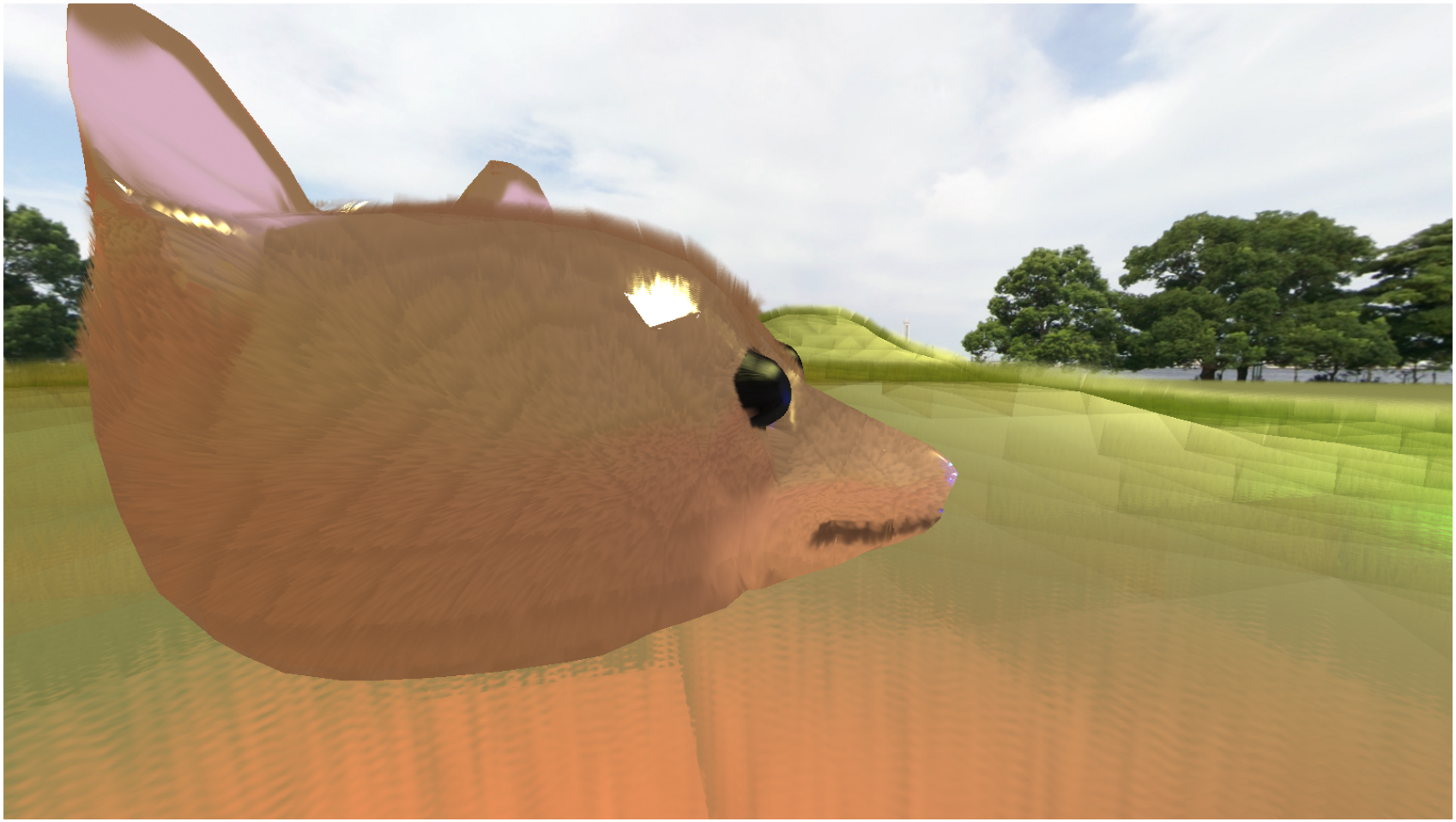
Blending

Hair and fur notably have very fine detail
My methods uses a lot of alpha.
Hard to depth-sort volumes and billboards?
Lots of self-occlusion

Self-occlusion



No depth mask



Weighted Blended Order-Independent Transparency

Multipass multichannel method.

1. Render Opaque Elements (skybox, bases)
2. Semitransparent render
   - 2.1 Occlude base color from background (colored glass filtering) (optional)
   - 2.2 Add lit color to accumulation buffer (depth and alpha weighted contribution)
   - 2.3 Add weight to accumulation buffer alpha
   - 2.4 Lower revealage (remaining transparency) by original alpha
3. Composite accumulation-buffer onto image
   - divide by accumulation-alpha to scale the average
   - alpha of composite layer = 1 - revealage

Blending results