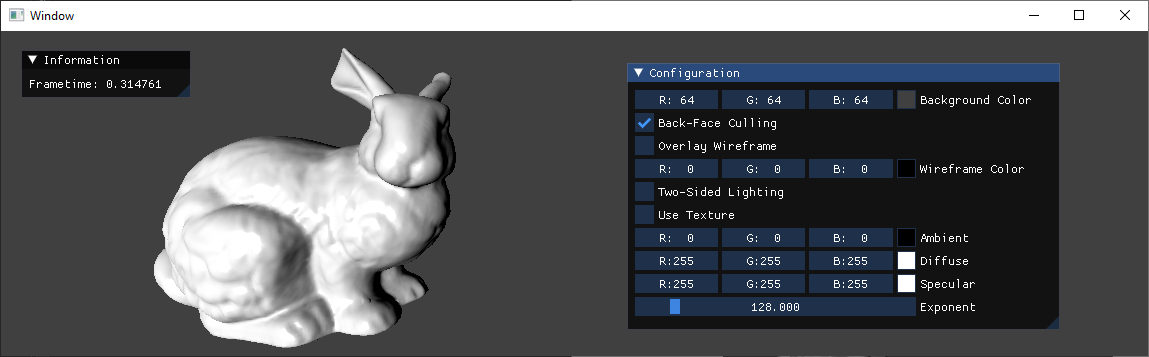
* + - * 1. Laden Sie das CBM Modell bunny.cbm unter Verwendung der Klasse CograBinaryMeshFile!
        2. Bestimmen Sie eine affine Abbildung , welche die Vertex Positionen normalisiert. Positionen eines beliebigen CBM Objektes sollen dabei auf einen quaderförmigen Bereich so abgebildet, dass
* der Schwerpunkt der Positionen im Ursprung des quaderförmigen Bereichs liegt,
* die längste Achse quaderförmigen Bereiches die Seitenlänge aufweist und
* die Seitenverhältnis der Raumachsen erhalten bleiben.

Nutzen Sie dazu möglichst viele glm und std Funktionen. Vervollständige Sie dazu f32m4 getNormalizationTransformation(f32v3 const\* const positions, ui32 nPositions).

* + - * 1. Erzeugen Sie einen Vertex- und Index-Buffer für das CBM Objekt. Erstellen Sie zudem einen Constant-Buffer um die nötigen Matrizen zur Transformation im Vertex-Shader bereitzustellen (Projektionsmatrix, Modelview-Matrix, etc.). Nutzen Sie dazu die Matrix , die glm Methoden perspectiveFovLH\_ZO<f32> sowie die Maustransformation m\_examinerController.getTransformationMatrix() um den Constant-Buffer zu füllen.

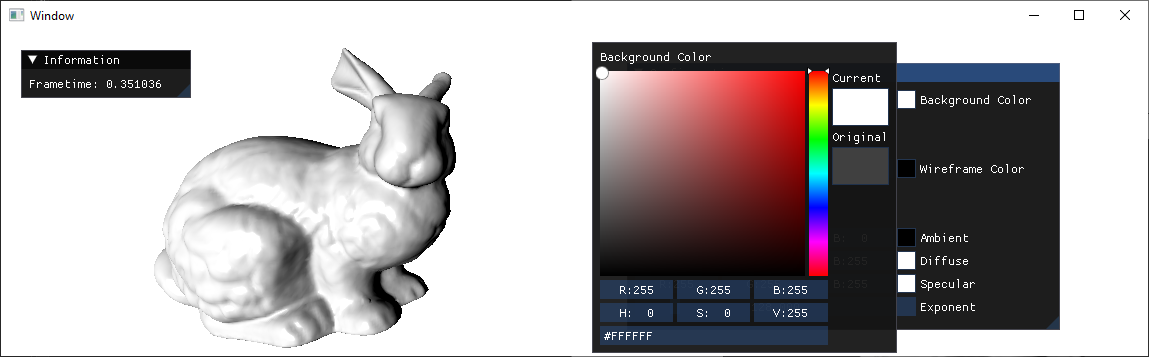
Passen Sie den Vertex-Shader an um das Modell darzustellen.

* + - * 1. Implementieren Sie folgendes UI:

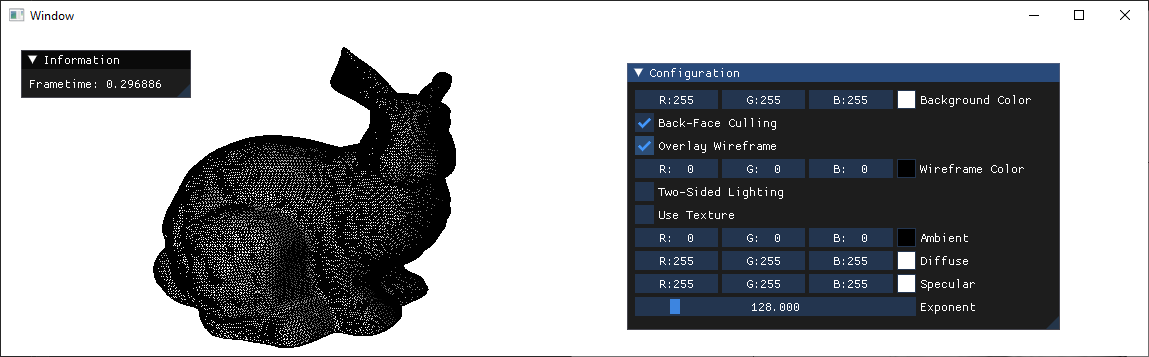


Folgendes Verhalten soll dabei umgesetzt werden:

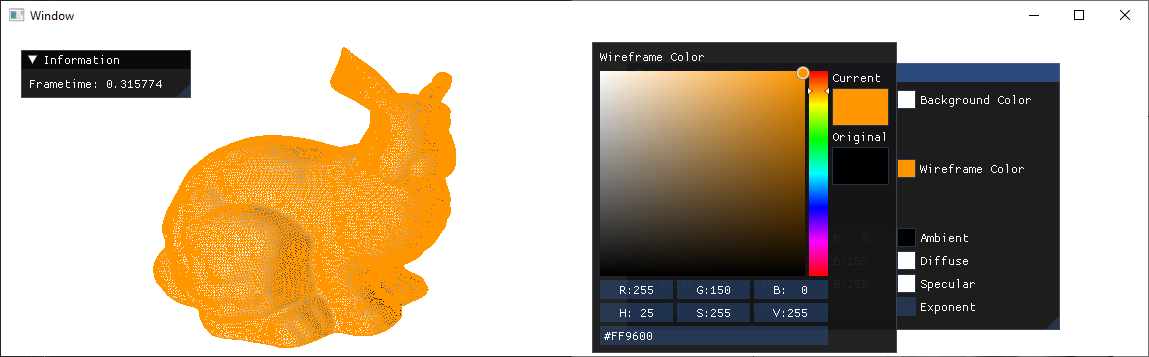
Background Color: Änderung der Hintergrund Farbe



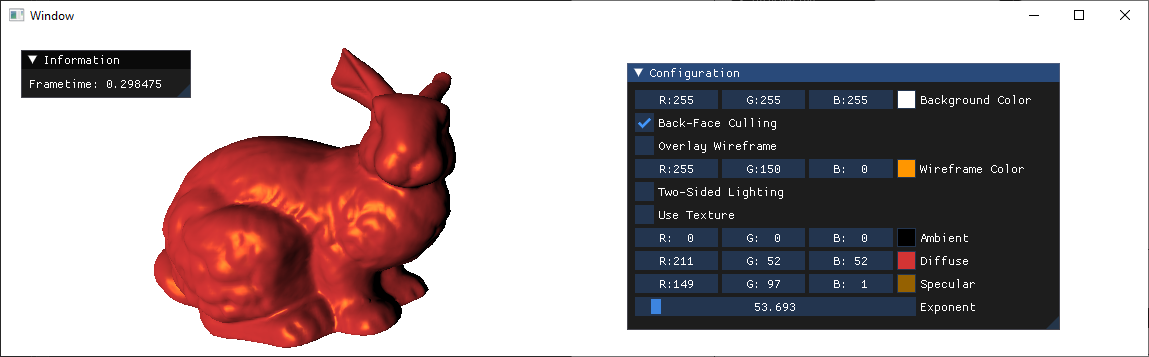
Overlay Wireframe: Drahtgittermodell überlagern an/aus



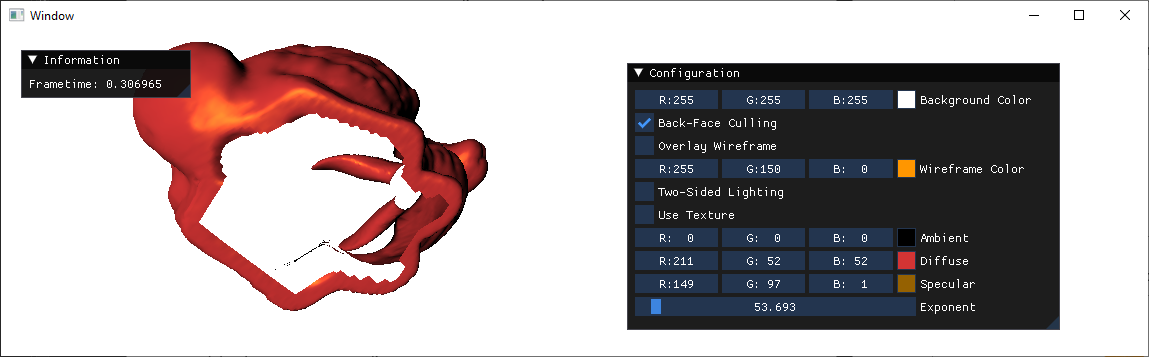
Wireframe Color: Drahtgittermodell Farbe einstellen



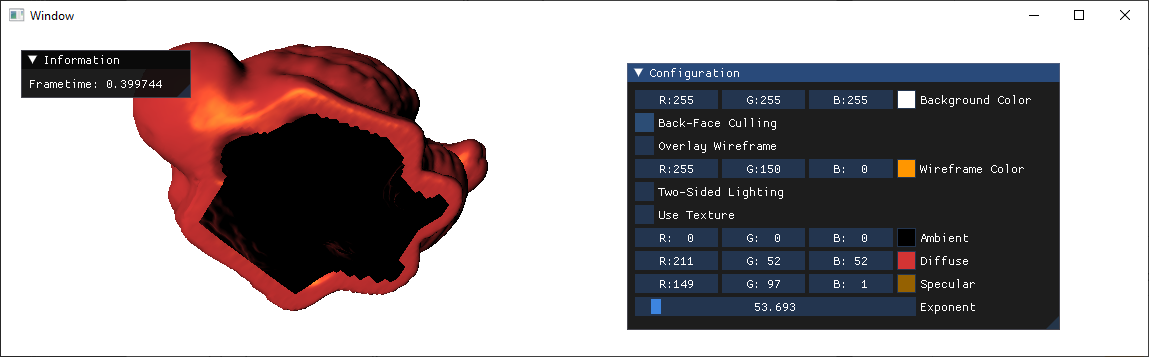
Ambient/Diffuse/Specular Color, Specular Expoent: Blinn-Phong Lighting Modell implementieren und von User:in konfigurieren lassen.



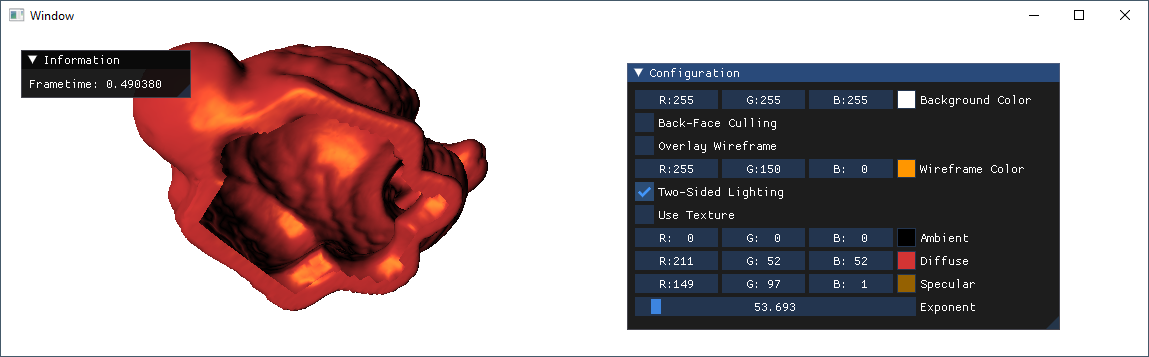
Backface Culling an



Backface Culling aus



Two-Sided Lighting



* + - * 1. Implementieren Sie geeignete weitere UI Elemente um Kamera- und und Lichtquelleparameter interaktiv anzupassen!

# Aufgabe 2

Laden Sie die Textur bunny.png und wenden Sie diese auf das Modell an.

