Inputs

- 3D model
  - Vertex coordinates (3D vectors)
  - Normals (3D vectors)
  - Edges, faces
  - Texture / UV coordinates (2D vectors)
- Unwrapped texture (2D image)

Algorithm

$I := \emptyset$
$P := \emptyset$
$T := \emptyset$
load image
simplify colors to available palette
separate colors
for each color c
    identify island borders $I_c = \left\{ I_{c_1}, I_{c_2}, ..., I_{c_m} \right\}$
    $I = I \cup I_c$
for each island i
    split border polygon p_i at edges into flat paths $P_i = \left\{ P_{i_1}, P_{i_2}, ..., P_{i_n} \right\}$
    $P = P \cup P_i$
    for each $f = 1, ..., k$
        compute intersections of parallel line L_f with border polygon p_i
        split L_f at edges into segments $F_{i,f} = \left\{ F_{(i,f)_1}, F_{(i,f)_2}, ..., F_{(i,f)_n} \right\}$
        $P = P \cup F_{i,f}$
for each path $p_j = \left(\left\{ P_1(u_1, v_1), P_2(u_2, v_2), ..., P_{o(u_o, v_o)} \right\}, c \right)$ in P
    interpolate 3D positions and get face indices from UV coordinates
        $V_1(x_1, y_1, z_1), f_1 \leftarrow P_1(u_1, v_1)$
        $V_2(x_2, y_2, z_2), f_2 \leftarrow P_2(u_2, v_2)$
        ...
    $T = T \cup \{(\{V_1, V_2, ..., V_o\}, f_o, c)\}$

Future considerations

- Input may be a side image of the duck $\rightarrow$ add a projection step at the beginning
- Output can be optimized, for example by sorting the segments to minimize distance or normal delta
- Segments may be non-linear (e.g. arcs) depending on robot interface and constraints
- Segments may need to be split when crossing sharp edges
- Other filling patterns can be implemented (e.g. concentric border, dots, waves, crosses, etc.)