Rendering a Coca-Cola Can with RenderMan

Xuwen Dong
Bournemouth University / NCCA
UK
s5803453@bournemouth.ac.uk

1 Introduction

This project recreates a Coca-Cola can using RenderMan 26.3 and the Python API. The object has a simple cylindrical form, but its realism depends on details such as the tapered rims, printed label, aluminium top, glossy coating, scratches, and reflections.

The work covers primitive and mesh modelling, physically based materials, texture and bump mapping, procedural surface variation, HDRI lighting, and camera depth of field.

Figure 1: Reference photograph of the real Coca-Cola can.

2 Real Object Analysis

The real can can be separated into several important visual features:

- Cylindrical body.
- Slight taper near the top and bottom.
- Rolled aluminium rim.
- Pressed top cap with circular grooves and pull-tab indentation.
- Printed red coating with white logo.
- Evidence of usage: slightly concave surface, dirt, and scratches.

3 Modelling with RenderMan Primitives

The modelling stage combined RenderMan primitives with generated polygon meshes. The can body was kept as a stable `Cylinder`, while the top and bottom rims were generated from profile rings using `PointsPolygons`. The top disk was built as a UV mesh so that bump detail could be controlled separately from the cylindrical label texture.

The first modelling pass was a simple blocking model made from a cylinder and disk shapes. This stage was used to find the basic relationship between the body height, body radius, top disk, and bottom disk before building more accurate rim geometry. After these proportions were established, the measured positions were reused to create more precise `PointsPolygons` profile meshes.

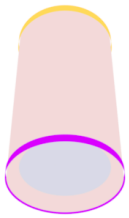


Figure 2: Initial geometry blocking using cylinder and disk primitives to establish the can proportions and position relationships.

Implementation summary:

- Body created with RenderMan `Cylinder`.
- Top/bottom rims generated from profile rings using `PointsPolygons`.
- Top disk generated as UV disk mesh for bump mapping.
- Multiple instances used for final composition.

3.1 Polygon Mesh Faceting

One key problem was faceting on the can top and taper area. In the first version, the highlight revealed a blocky surface because the profile had too few rings and the radius changed too abruptly.

Listing 1: Initial profile with abrupt taper

```
profile = [
    (0.200, 0.310),
    (0.180, 0.350),
]
```

The solution was to add more vertical profile rings so that the taper changed gradually. This produced smoother silhouettes and more continuous highlights.

Listing 2: Smoothed taper profile

```
profile = [
    (0.200, 0.310),
    (0.198, 0.320),
    (0.195, 0.330),
    (0.191, 0.340),
    (0.186, 0.347),
    (0.180, 0.350),
]
```

RenderMan subdivision surfaces using Catmull-Clark subdivision were also tested. This approach could produce a smoother can shape, but it required much cleaner topology than the polygon version. When the profile rings overlapped or created very thin quads, RenderMan reported errors such as `Z10004 invalid bounds` or `Instance has invalid bounds`. The cause was unstable topology, usually from overlapping rings, very thin polygons, identical `z` values, or degenerate quads.

Listing 3: Degenerate rings with the same height

```
(0.200, 0.370),
(0.195, 0.370),
```

To avoid this, the profile was adjusted so that each ring had a clear height difference and a less abrupt shape transition.

Listing 4: Safer profile transition

```
(0.200, 0.358),
(0.198, 0.365),
(0.195, 0.368),
(0.190, 0.366),
```

The final modelling workflow therefore used a hybrid approach: `Cylinder` for the main body, `PointsPolygons` profile meshes for the rims, and a UV disk mesh for the top cap. This was more stable than modelling the whole can as one subdivision mesh.

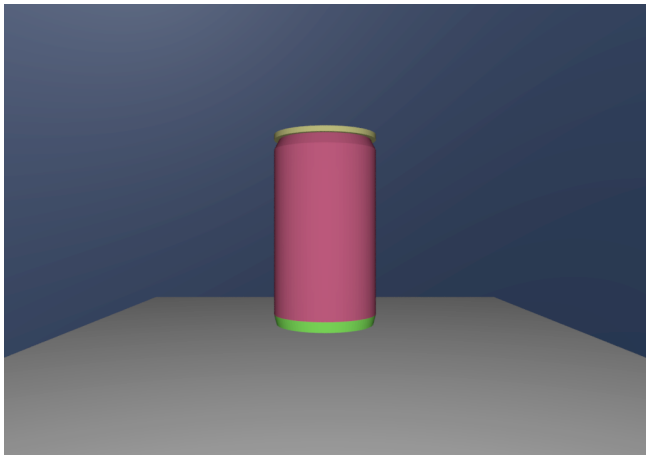


Figure 3: More accurate can geometry built from the initial blocking data using `PointsPolygons` profile meshes.

4 BRDF and Material Design

The can body is painted aluminium with a red printed coating and a glossy clear surface. The top and bottom are exposed aluminium, with rougher, brushed metallic reflection.

The material implementation used:

- `PxrSurface` used for body and metal parts.
- Body uses diffuse label texture plus specular reflection.
- Aluminium uses metallic Fresnel parameters, roughness, and bump detail.

5 Texture and Bump Detail

The texture stage focused on three surface types: the printed Coca-Cola label, the pressed top cap, and the wooden ground plane.

- Coca-Cola label texture wrapped around cylinder.
- Top cap bump texture used for pressed circular grooves.
- Wood texture used for ground plane, referenced from `American_walnut_pxr` in Pixar's tiling texture library [2].

Figure 4: Upscaled Coca-Cola label texture used for the cylindrical body.

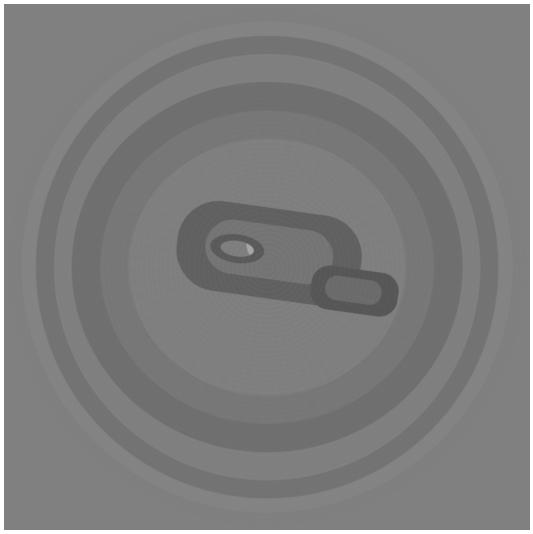
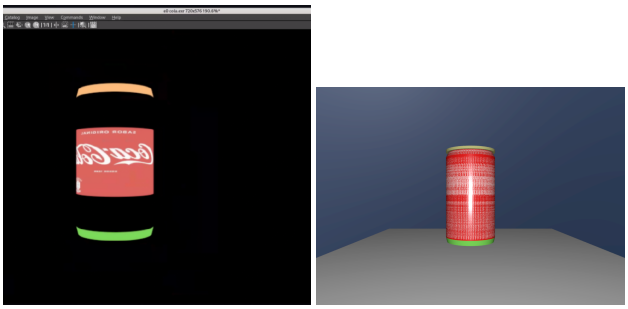


Figure 5: Radial top-cap bump texture prepared for the disk mesh.

5.1 Cylinder Label UV Direction

The first label mapping tests showed several UV problems: the label could appear flipped left-to-right, vertically inverted, repeated horizontally, or compressed around the can. The final mapping used `PxrManifold2D` to control the direction and placement of the label on the cylinder.



(a) Offset and flipped label. **(b) Repeated body texture.**

Figure 6: Early cylindrical UV mapping problems during label placement.

Listing 5: Label manifold settings for the cylindrical can body

```
ri.Pattern(
    "PxrManifold2D",
    "label_manifold",
    {
        "float scaleS": [-1.0],
        "float offsetS": [1.0],
        "float scaleT": [0.45],
        "float offsetT": [0.3],
        "int invertT": [1],
    },
)
```

The negative `scaleS` and positive `offsetS` corrected the horizontal direction, while `scaleT`, `offsetT`, and `invertT` were used to fit the label vertically on the can body.

5.2 Top Cap Bump UV

The top cap bump was also problematic. Early tests produced dark, stretched, or distorted patterns because a disk primitive uses a polar-style UV layout, with the center compressed into a small UV region. A normal rectangular bump texture therefore stretched toward the disk center.

The solution was to redraw the bump as a radial texture specifically designed for the disk UV layout. This made the circular grooves and pull-tab indentation align with the geometry more naturally.

(a) Distorted disk UV. **(b) Corrected radial bump.**

Figure 7: Top cap bump before and after redrawing the texture for disk UVs.

6 Procedural Noise and Wear

Real cans are not perfectly clean: paint has micro roughness, metal has scratches, and reflections are slightly broken. Without procedural variation, the render looked too clean and "perfect CG".

The implementation used:

- OSL procedural scratch/noise pattern used for surface variation.
- Noise connected to roughness or bump to avoid perfect CG surfaces.
- Subtle values were used to keep the object believable.

6.1 OSL Scratch Integration

An OSL scratch shader was used to generate fine procedural variation across the surface.

Listing 6: OSL procedural scratch shader

```
shader scratch(
    float scale = 80,
    float strength = 0.25,
    output float result = 0
)
{
    point PP = transform("shader", P) * scale;

    float line = noise(
        "uperlin",
        point(PP[0], PP[1] * 0.08, PP[2])
    );
```

```

    result = 0.18 + line * strength;
}
```

The scratch output was connected to material roughness so that reflections were subtly broken up.

Listing 7: Connecting scratch output to roughness

```
"reference float specularRoughness":
    ["scratch_roughness:result"]
```

For additional surface detail, the same procedural pattern could also drive a small bump signal.

Listing 8: Connecting procedural scratches to bump

```
ri.Pattern(
    "PxrBump",
    "body_scratch_bump",
    {
        "reference float inputBump":
            ["body_scratch:result"],
        "float scale": [0.002],
    },
)
```

7 Lighting and Environment Map

The scene used an environment-driven lighting setup:

- HDRI / dome light used for natural outdoor reflections.
- Wooden ground plane gives contact shadow and scale.
- Reflections on the can are strongly affected by the environment.

Before the final environment setup, a simple daylight render was used to check the model, material response, and cast shadow. The later HDRI test provided more varied reflections on the aluminium rim and glossy red coating, which made the can feel less flat.

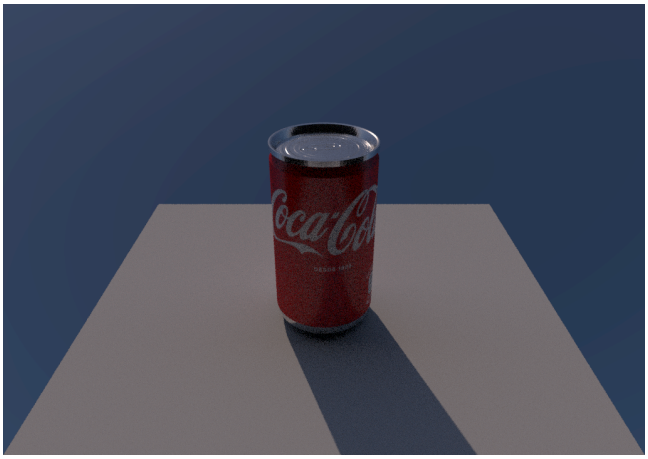


Figure 8: Daylight render used to test the model, material response, and shadow direction.

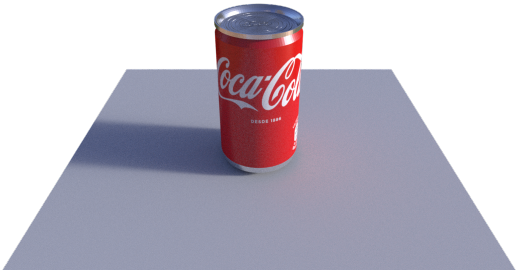


Figure 9: HDRI lighting test with stronger environmental reflection on the can surface.

8 Camera Effects

The camera was set up to resemble a product photograph. The depth of field and background separation are most visible in Figure 11.

- Depth of field used to create photographic focus.
- Background blur helps separate the object from the environment.
- Final images rendered at 1080p.

9 Results

The final renders show the completed can shader, geometry, label mapping, surface wear, lighting, and camera depth of field.

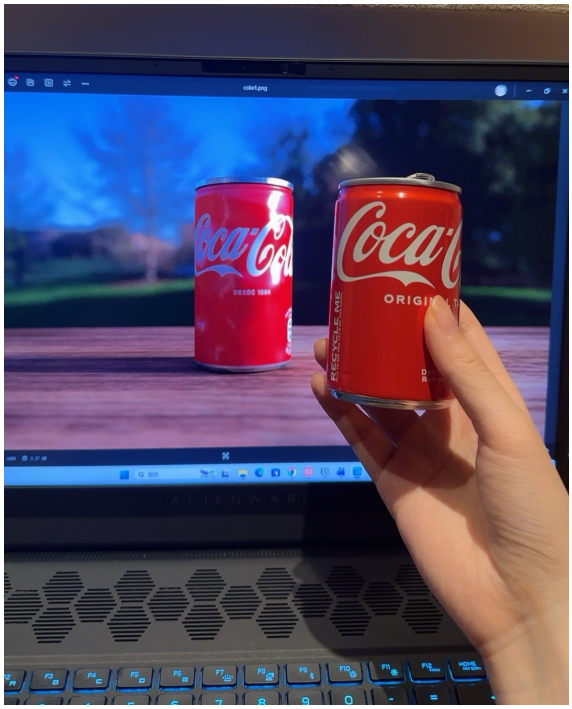


Figure 10: Comparison between the real Coca-Cola can and the simulated RenderMan result.

Figure 11: Final render 1: standing can product shot.

Figure 12: Final render 2: two-can composition on wood surface.

10 AI Usage

ChatGPT by OpenAI was used to help translate and organize the report into LaTeX format [1]. AI-assisted processing was also used to upscale the body texture and support planning for smoother profile geometry. The final modelling, shader setup, texture placement, parameter choices, and evaluation were completed manually by the author.

11 Limitations and Future Work

The project could be improved in three main areas:

- The AI-generated top-cap bump should be manually refined for more accurate groove depth, edge sharpness, and pull-tab placement.
- The red painted aluminium needs stronger metallic sparkle and clearer specular variation, possibly with anisotropic reflection.
- The surface is still too clean; future work could add procedural dirt, dust, fingerprints, and small stains.

12 Conclusion

The final project combined simple primitive modelling with more detailed mesh and shading work. The pipeline can be summarized as:

Listing 9: Final rendering pipeline

```
Cylinder
+ profile mesh rims
+ UV disk top
+ PxrSurface BRDF
+ texture mapping
+ procedural OSL scratches
+ HDRI lighting
+ DOF camera
+ wood textured ground
```

This workflow produced a realistic Coca-Cola can product render while keeping the geometry stable in RenderMan. The main technical lessons were to avoid abrupt mesh profiles, prevent degenerate topology, control UV direction explicitly, design disk bump maps for polar-style UVs, and use procedural variation to reduce the clean CG appearance.

References

[1] OpenAI. 2026. ChatGPT. https://chatgpt.com/ AI assistance used for translation, LaTeX report organization, and texture/geometry workflow support; accessed 2026-05-27.

[2] Pixar Animation Studios. 2018. Pixar One Twenty Eight. https://renderman.pixar.com/pixar-one-twenty-eight Tiling texture library, accessed 2026-05-27.