

# 3DCG Illustrations [2026/06/01]

Non-Research Tips for Information Science Researchers

情報科学研究補助技法



**Yuki Koyama**  
**小山 裕己**

<https://koyama.xyz/>

- 2017:  
Ph.D. from **UTokyo** (Computer Science)
- 2017—2025:  
Researcher (Senior Researcher; 2022–)  
at **AIST (産業技術総合研究所)**
- 2021—Current:  
Technical Adviser at **Graphinica, Inc.**  
(an anime production company using 3DCG)
- 2025—Current:  
Associate Professor  
at **UTokyo** (Precision Engineering)



**Motivation: Why 3DCG?**



In general, **good illustrations** ...

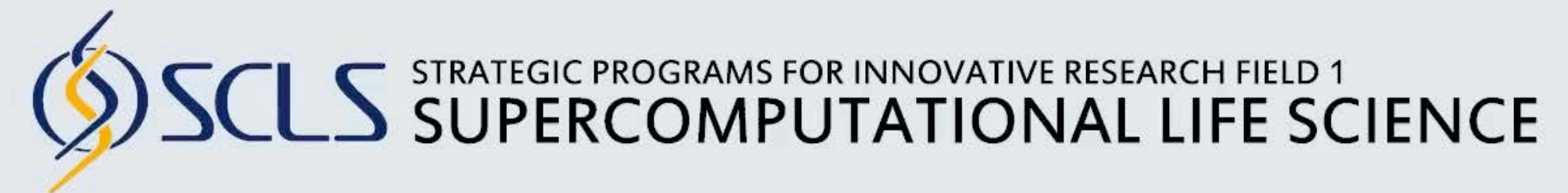
1. help **audience better understand** your research  , and
2. increase presentation appeal  (→ **buzz**).

**3DCG** is (sometimes) an effective technique for these purposes.



# 1. Better Understanding

- Processed faster by the brain, **making it easier for audiences to grasp information**
- Provides visual representations that **make complex ideas and data more accessible**



Japan

<https://www.youtube.com/watch?v=2LPboySOSvo>

## 2. Presentation Appeal

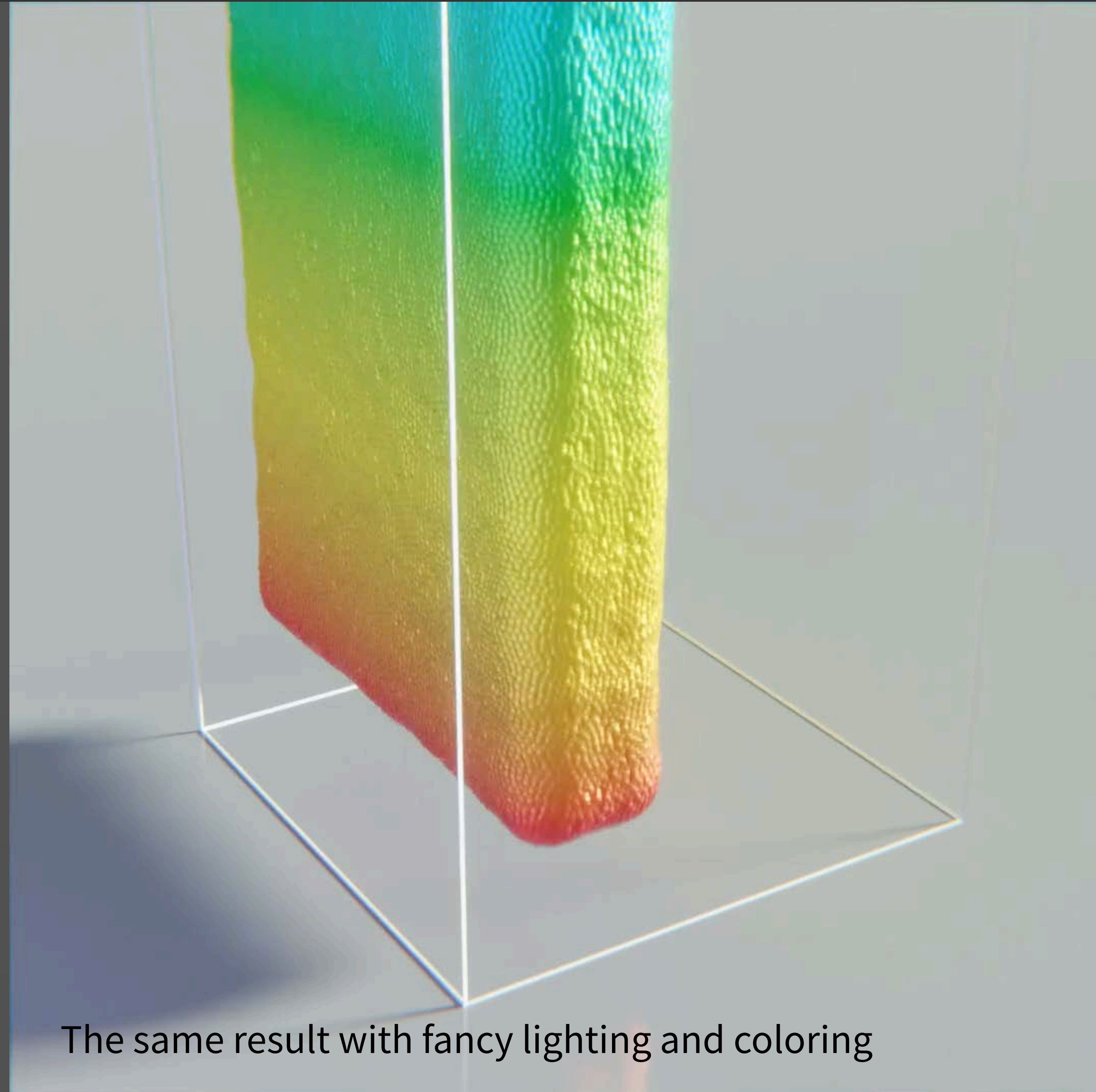
- Make presentations more memorable, leading to **better communication**
- Capture the audience's **attention**



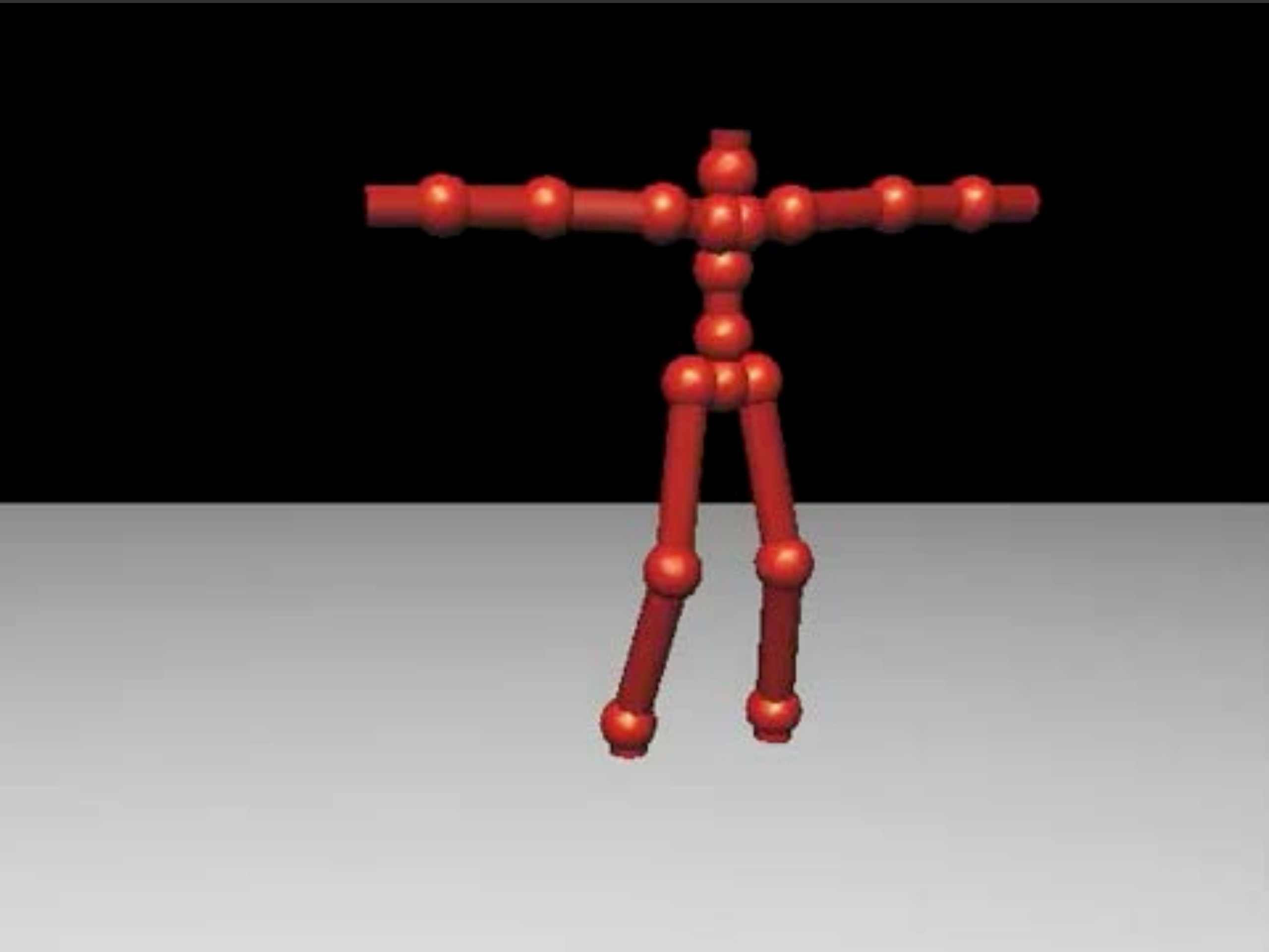
# Which is better?



Particle-based fluid simulation result, showing each particle as a black dot



The same result with fancy lighting and coloring



Which do you want to retweet?

**Scenarios: When to Use 3DCG?**



# Illustrating Complex Concepts

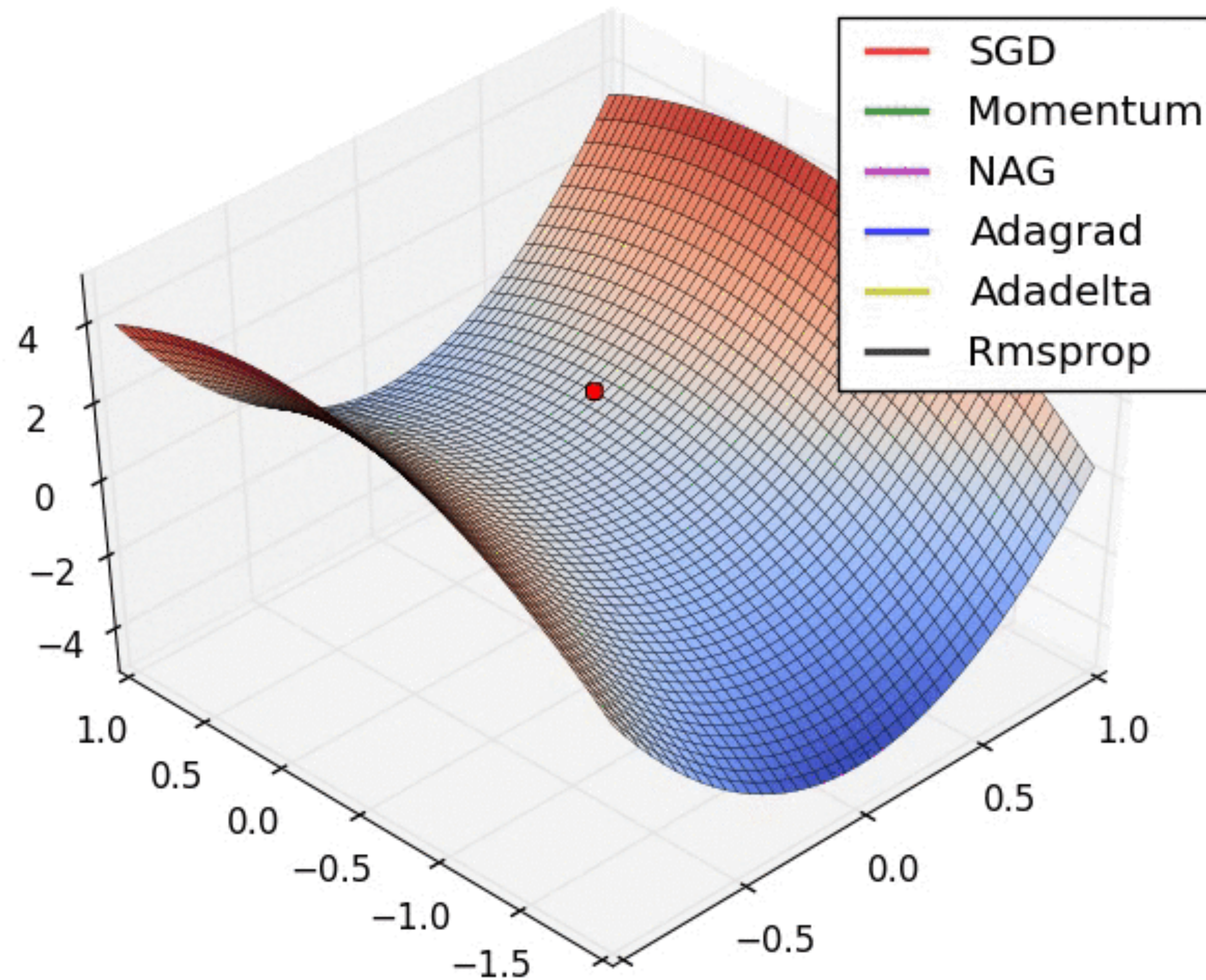
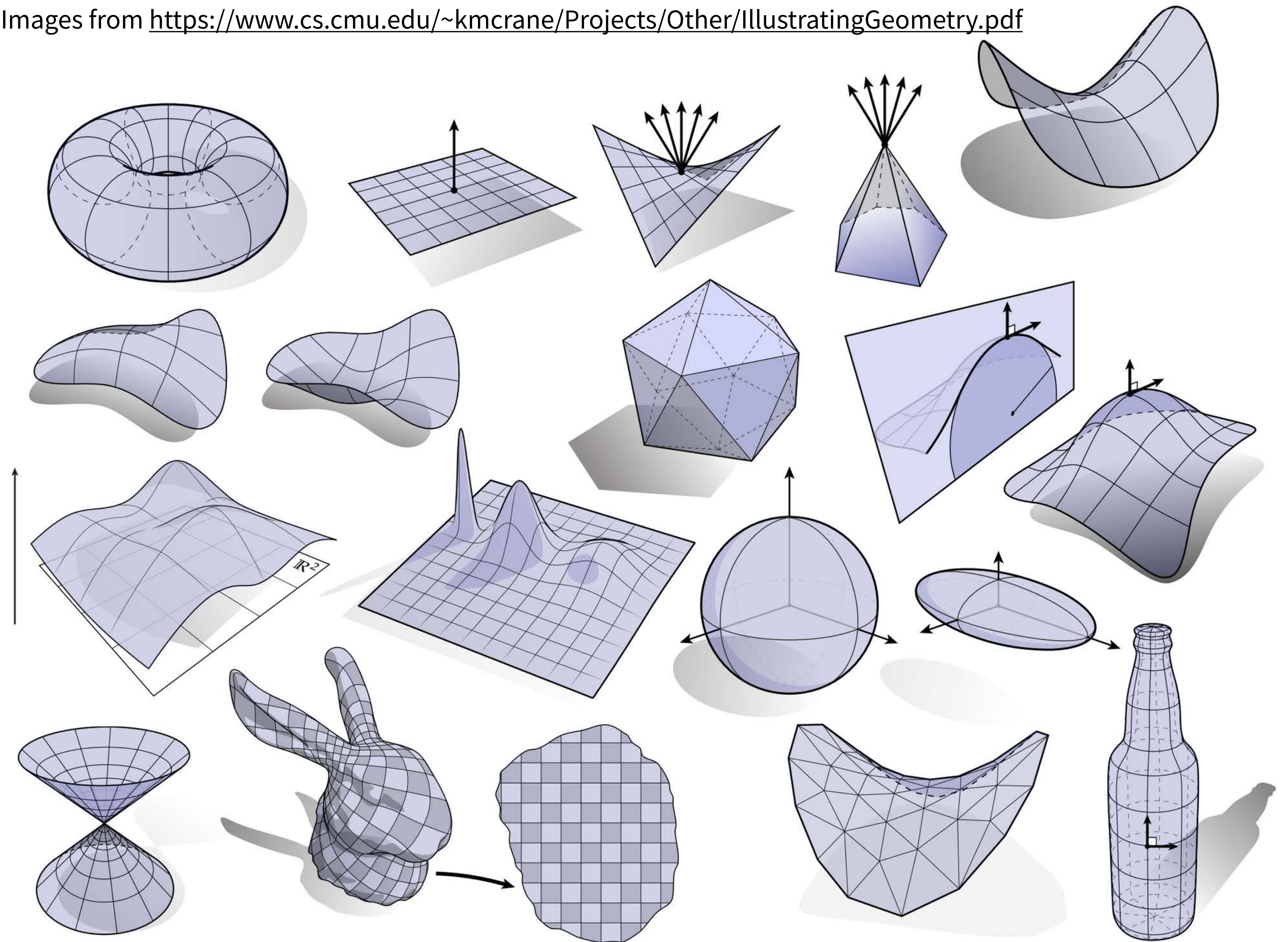


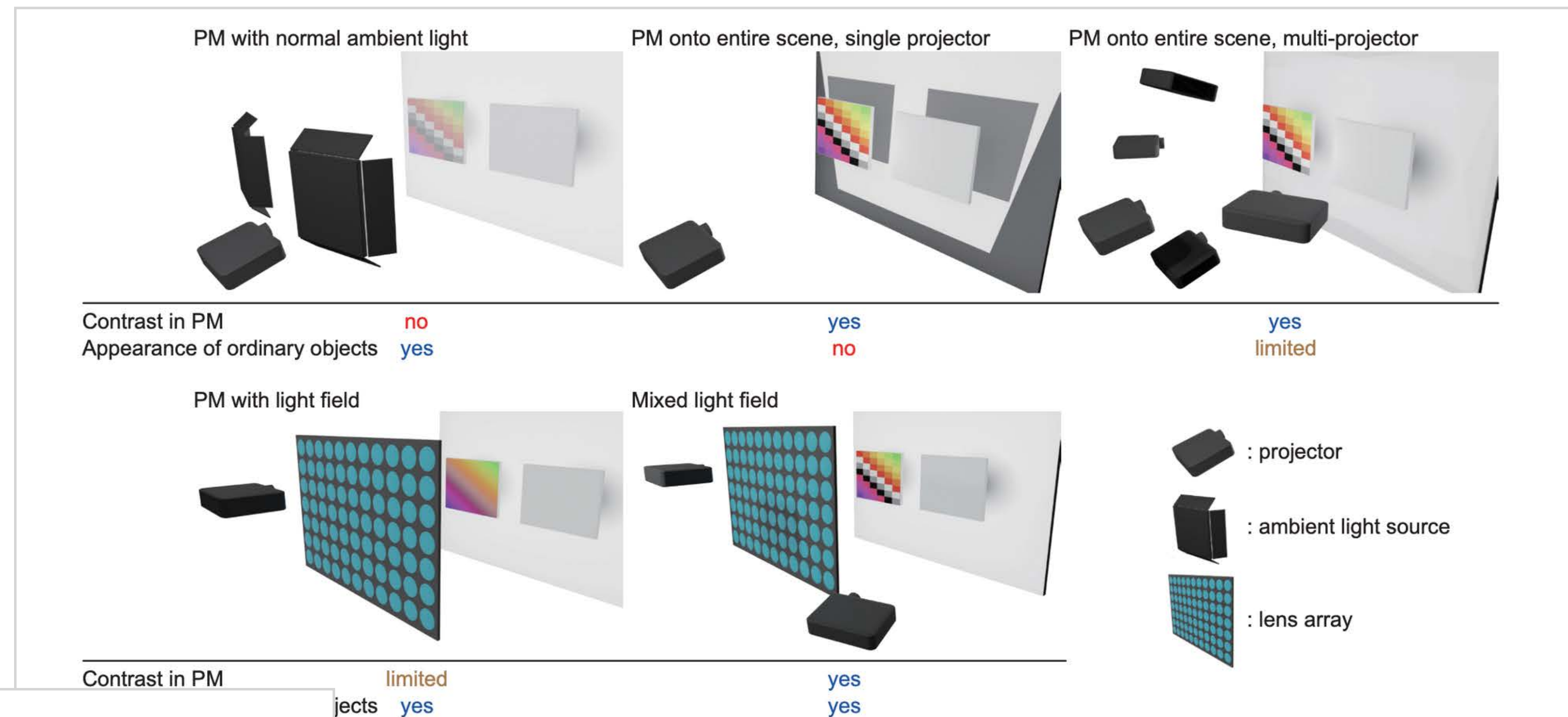
Image from <https://www.ruder.io/optimizing-gradient-descent/>

# Illustrating Complex Concepts

Images from <https://www.cs.cmu.edu/~kmc Crane/Projects/Other/IllustratingGeometry.pdf>



# Illustrating Complex Experimental Settings



explored to realize PM with brightly lit surroundings. The test setup involves the placement of two small boards within the target, and the other represents an ordinary object. A larger board is positioned behind them. The proposed concept, aims to achieve a high level of contrast in PM presentations while simultaneously reproducing the natural appearance

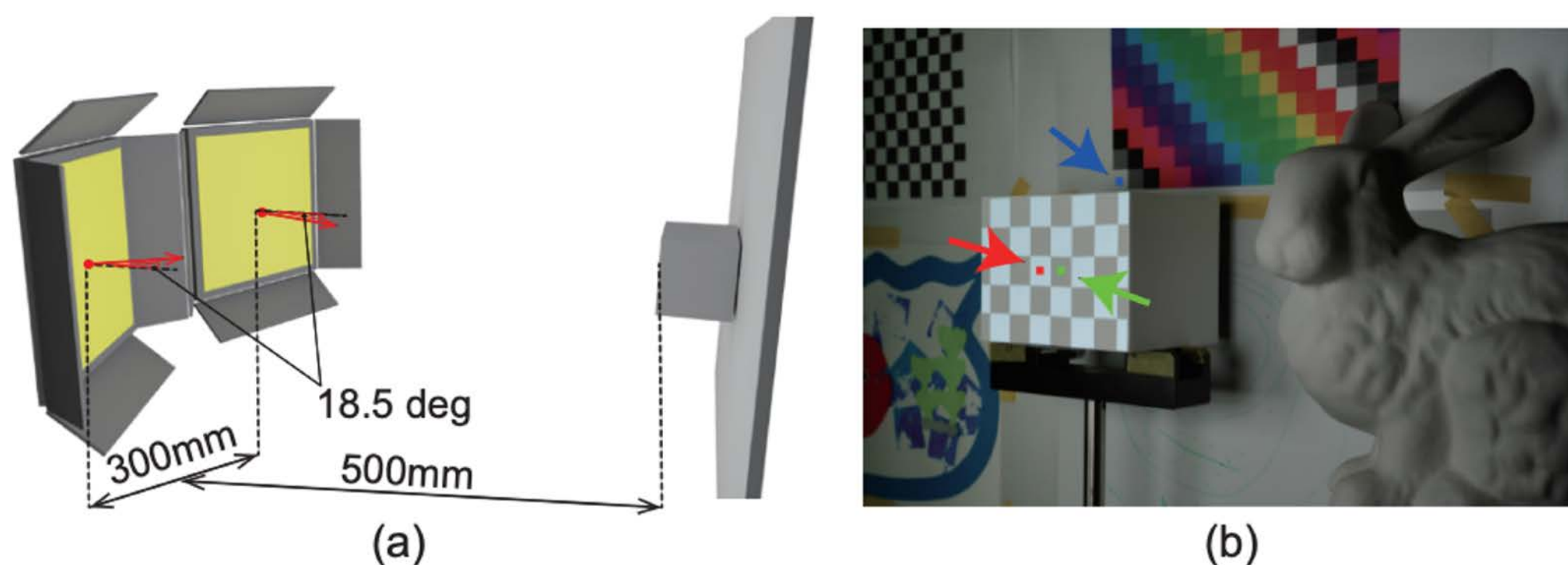
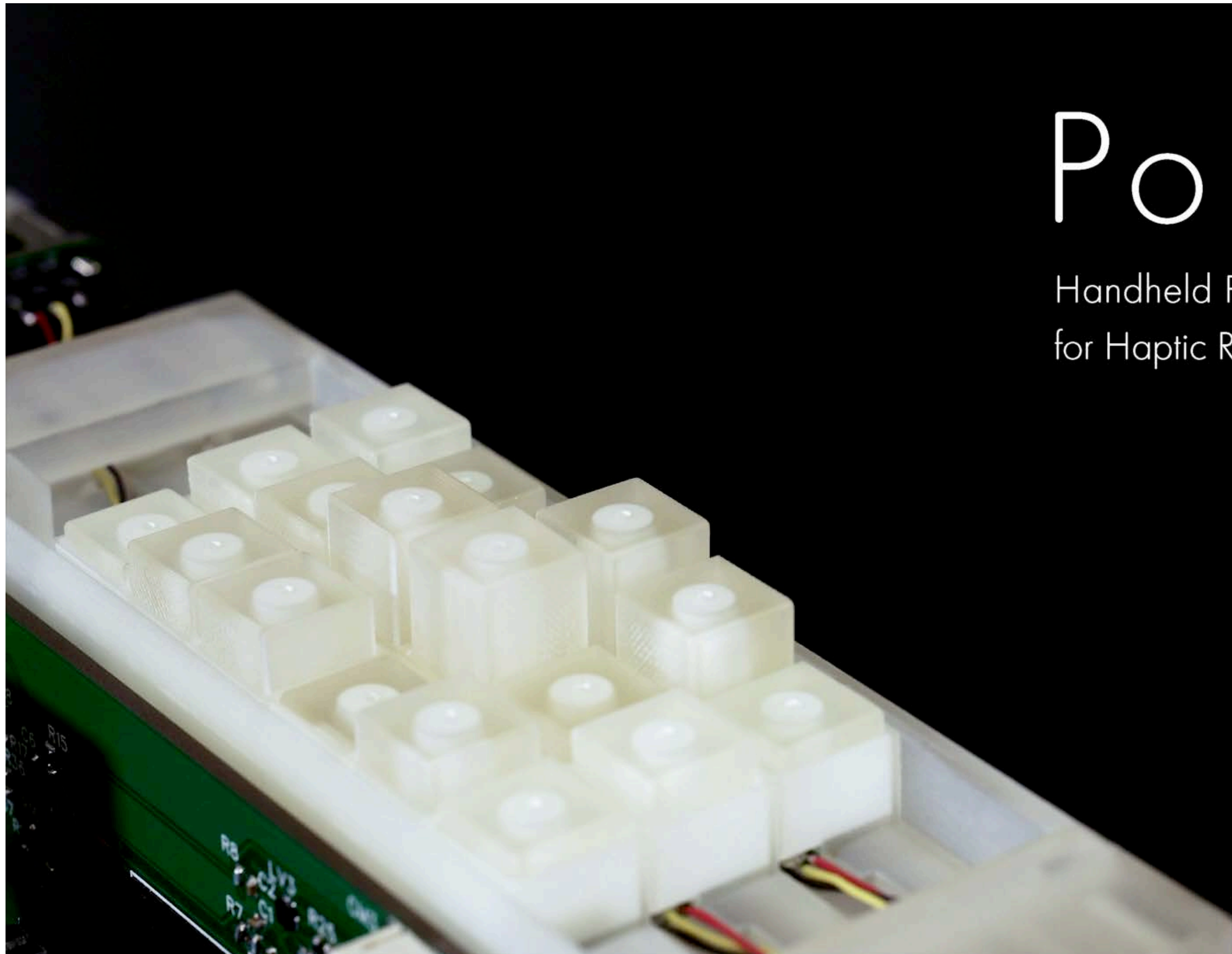


Fig. 12: (a) Arrangement of panel lights in the scene. (b) Measured point for the irradiance and contrast evaluations. *White, black, and background* values were measured at the red, green, and blue points, respectively.

# Mechanical Assemblies: Step-by-Step Assemble




PoCoPo

Handheld Pin-based Shape Display  
for Haptic Rendering in Virtual Reality

The University of Tokyo

Shigeo Yoshida\*  
Yuqian Sun\*  
Hideaki Kuzuoka



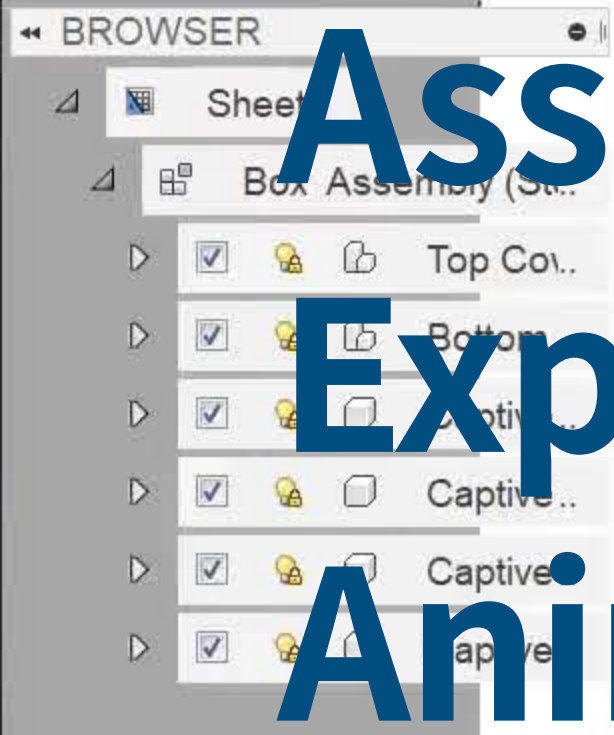
\*The first two authors contributed equally to this work.

# Mechanical Assemblies: Exploded View Animation

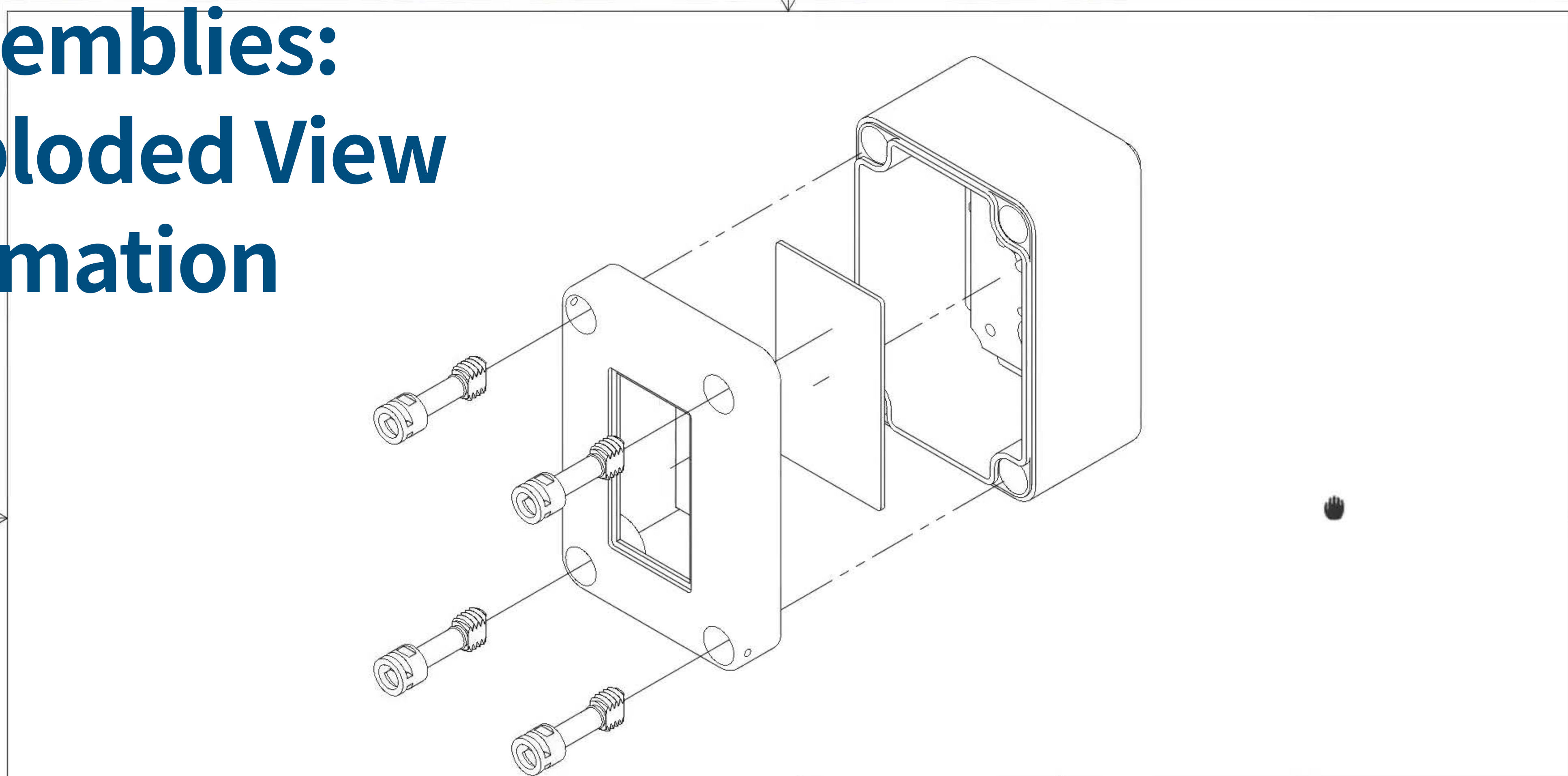
- Help understanding of the assembly, and
- Simply attractive



The 3D model was provided by George Kayesi for tutorial purposes  
<https://www.youtube.com/watch?v=IC0-9b0Rv2g>

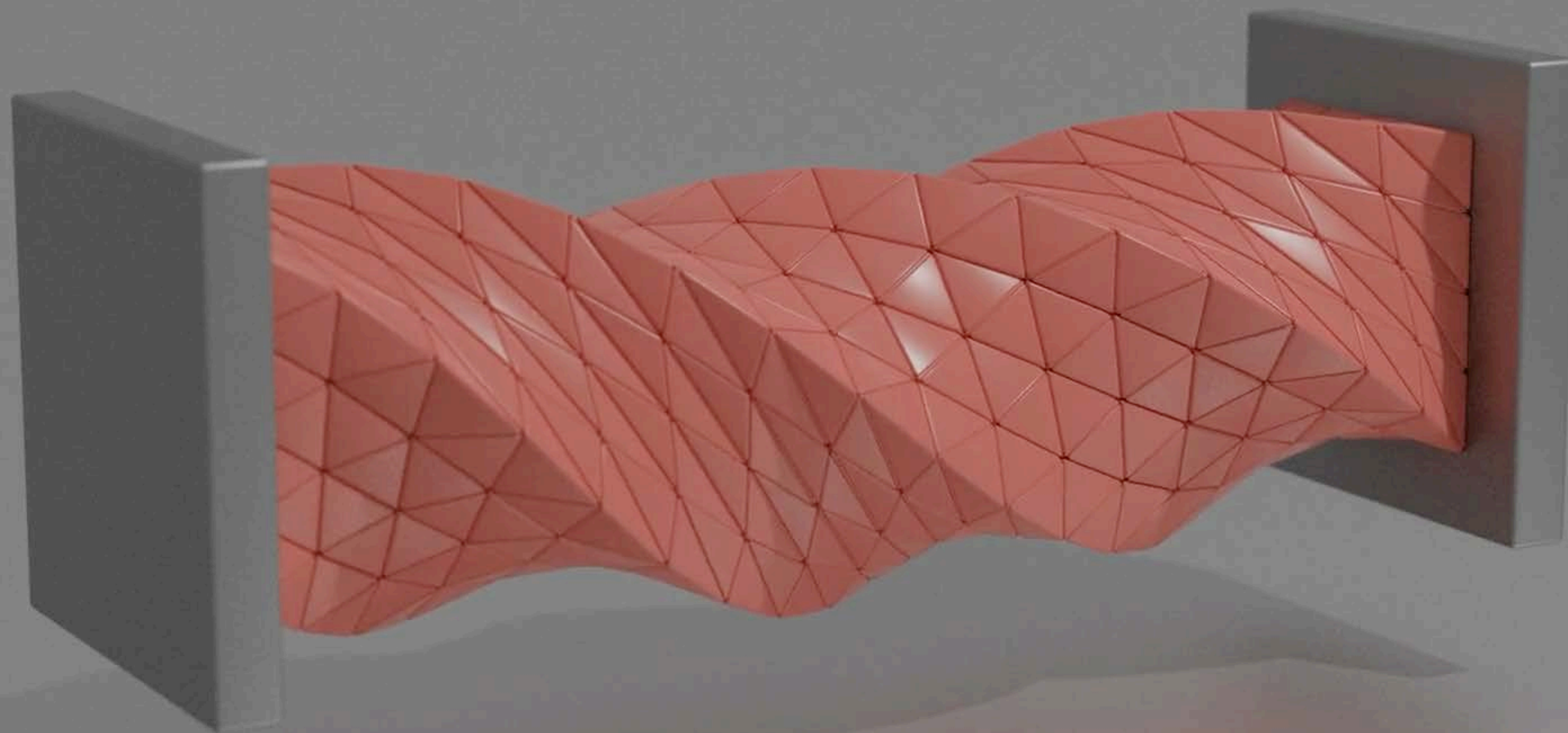


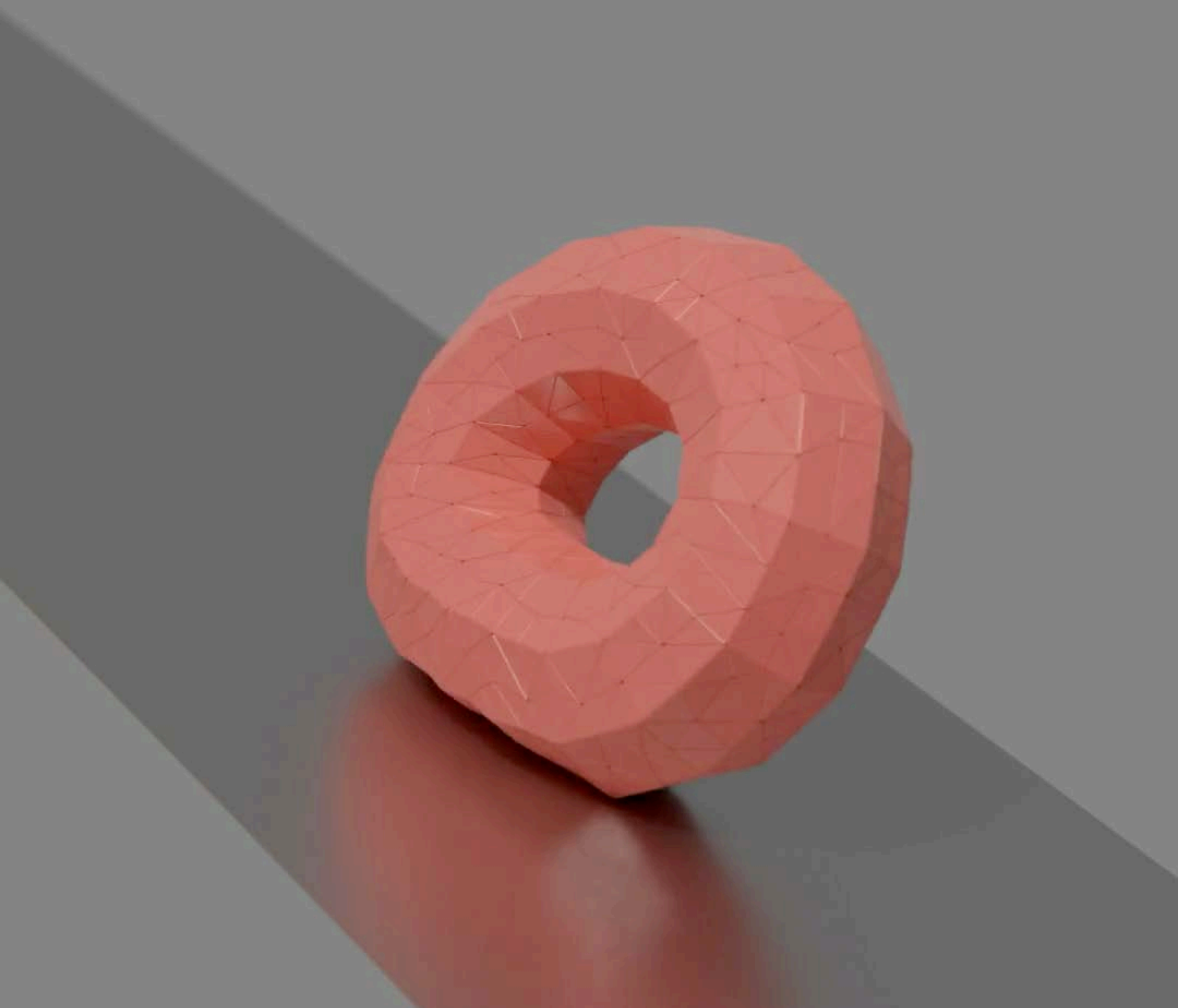
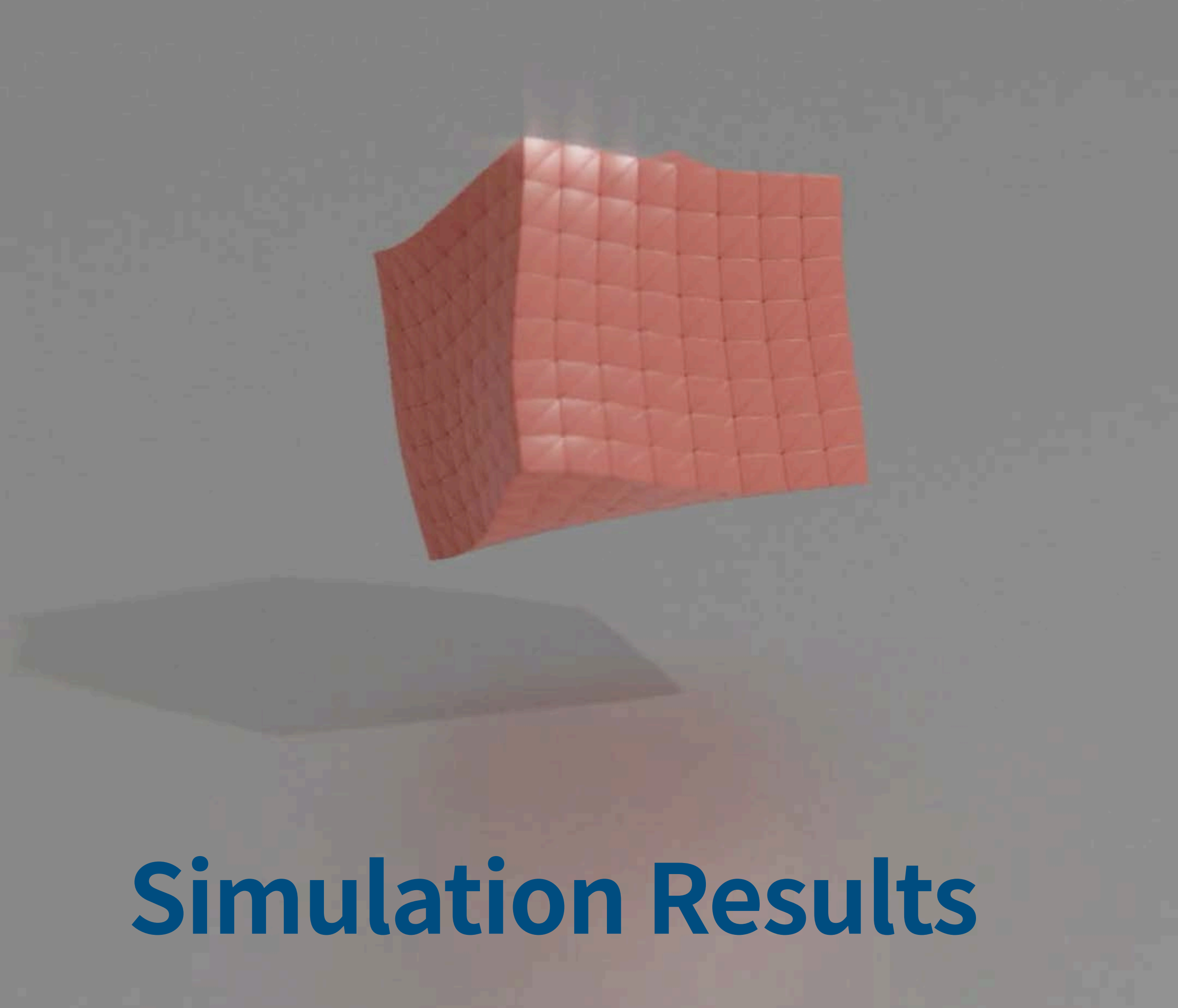
# Mechanical Assemblies: Exploded View Animation



PROJECT	My First Project
TITLE	Box Assembly

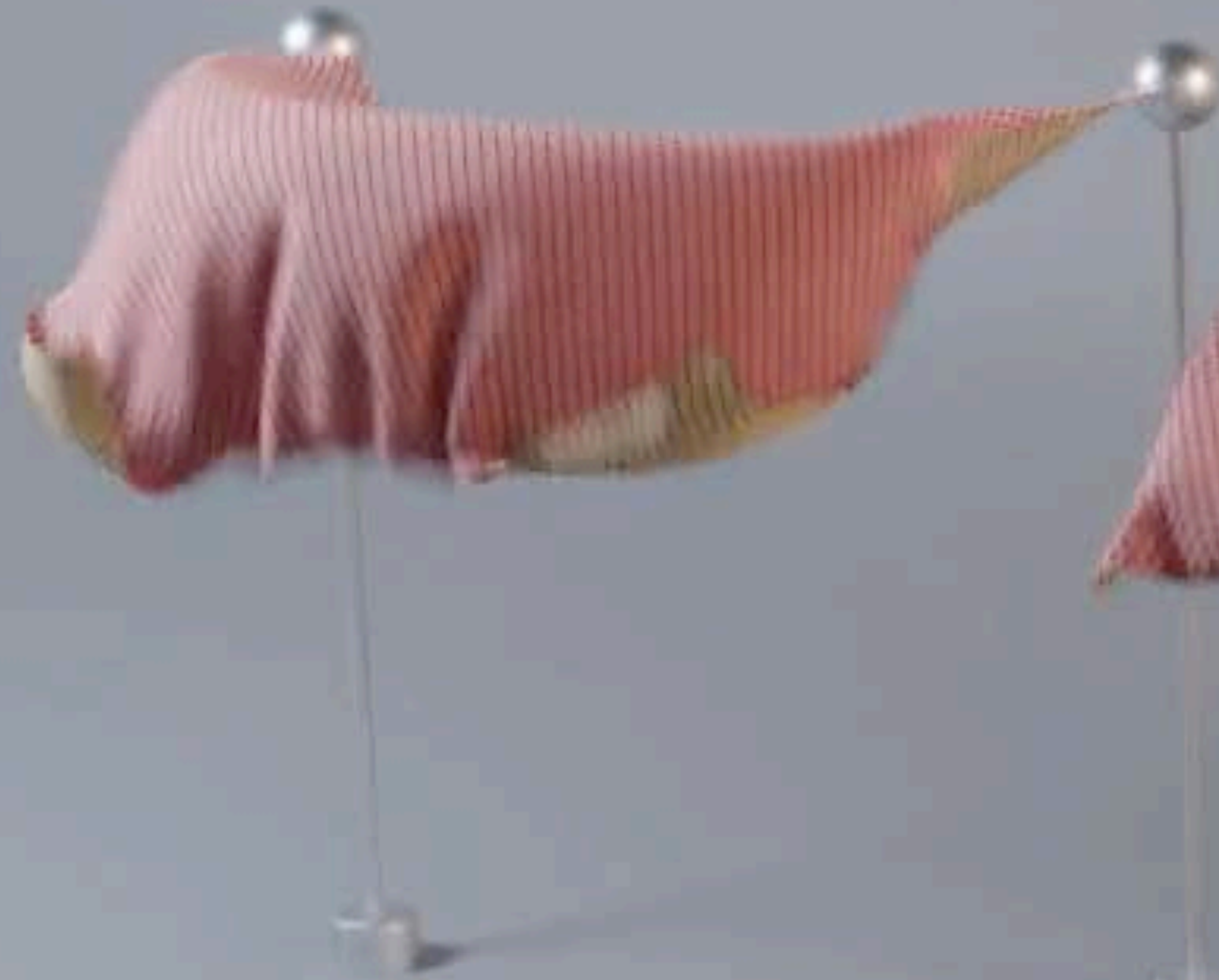
# Simulation Results



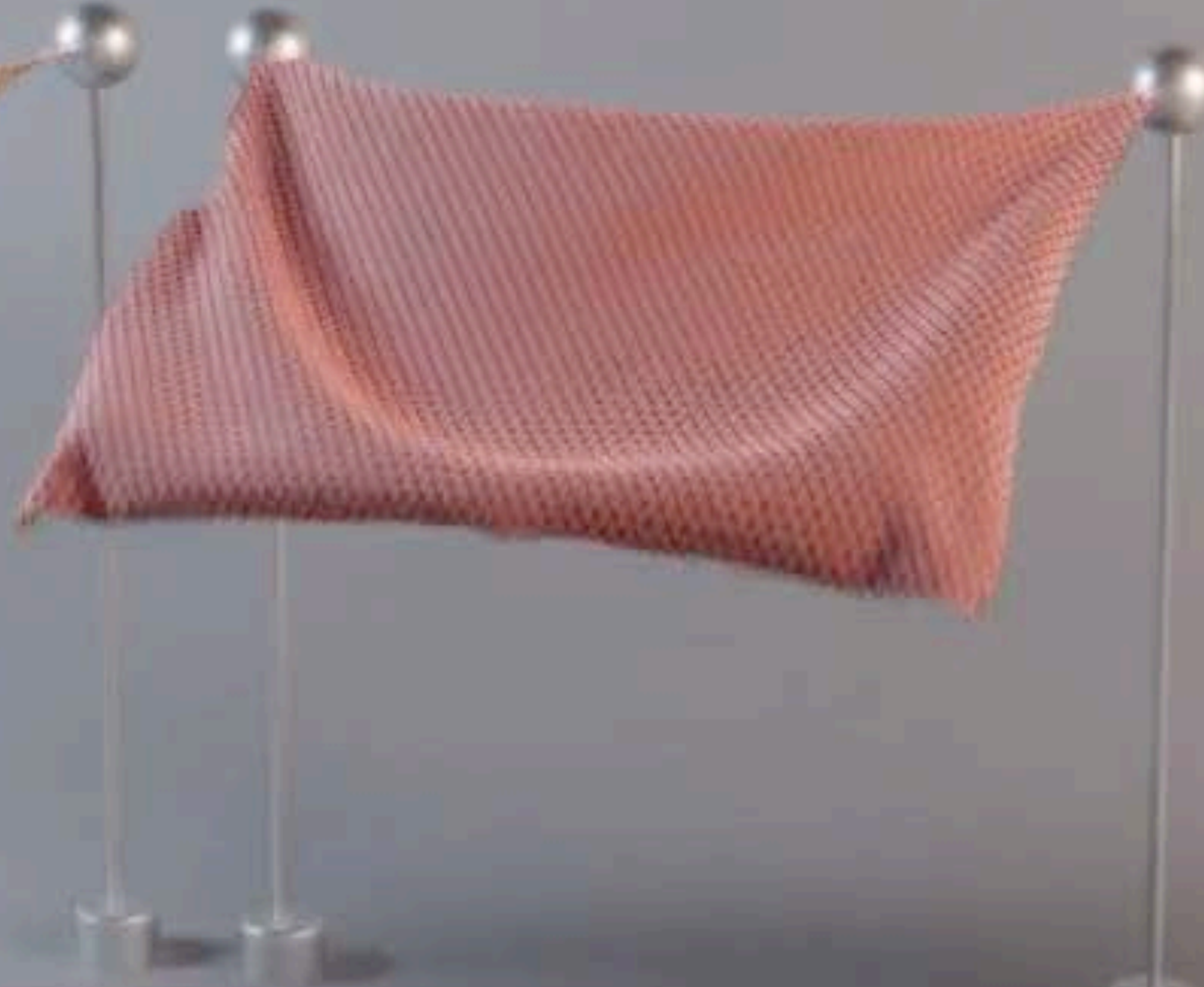


# Simulation Results

Small drag force  
Large lift force



Medium drag force  
Medium lift force

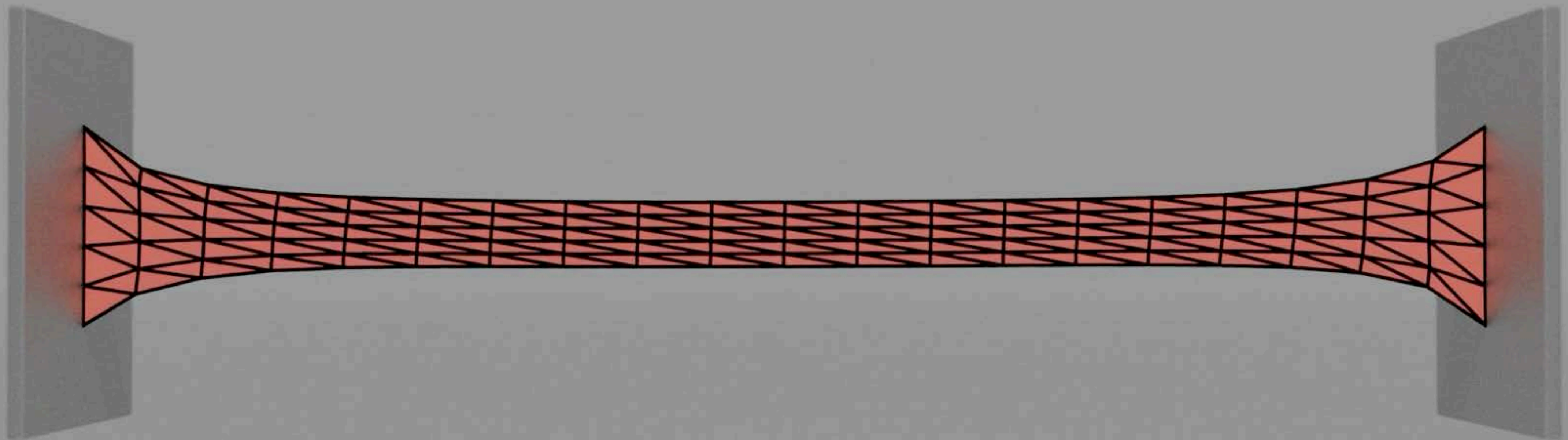


Large drag force  
Small lift force



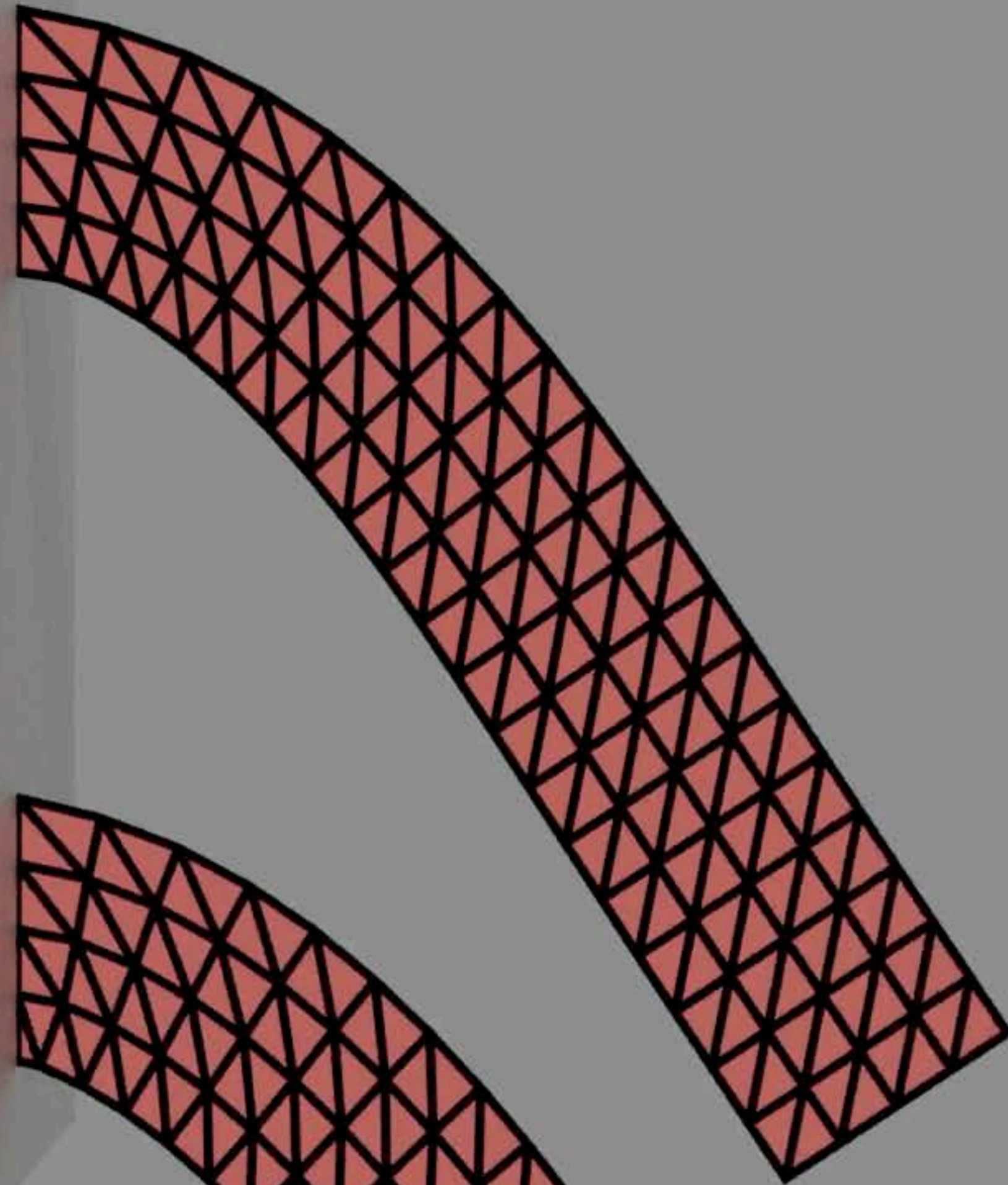
## Simulation Results

# Simulation Results (even 2D!)

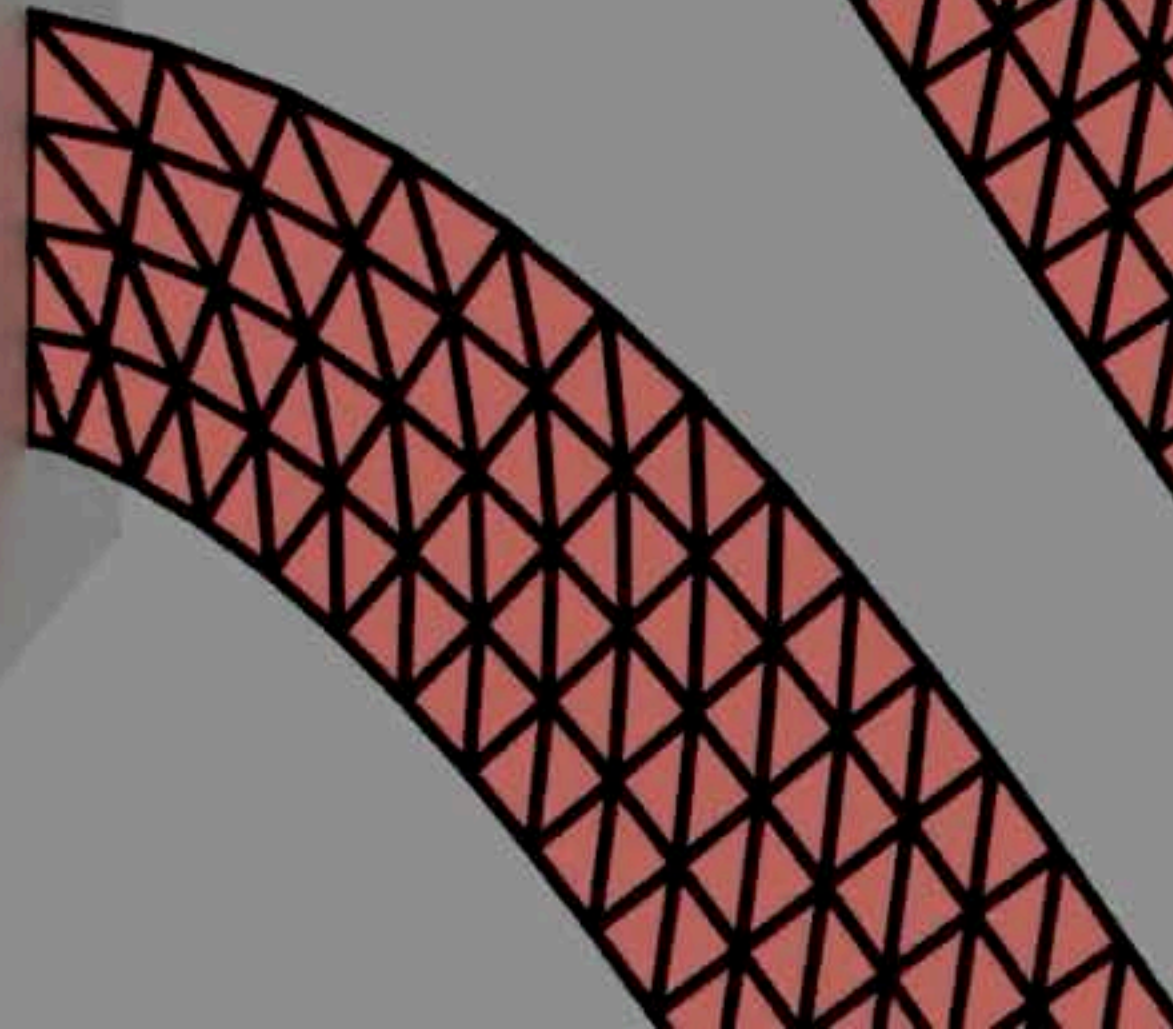


# Simulation Results (even 2D!)

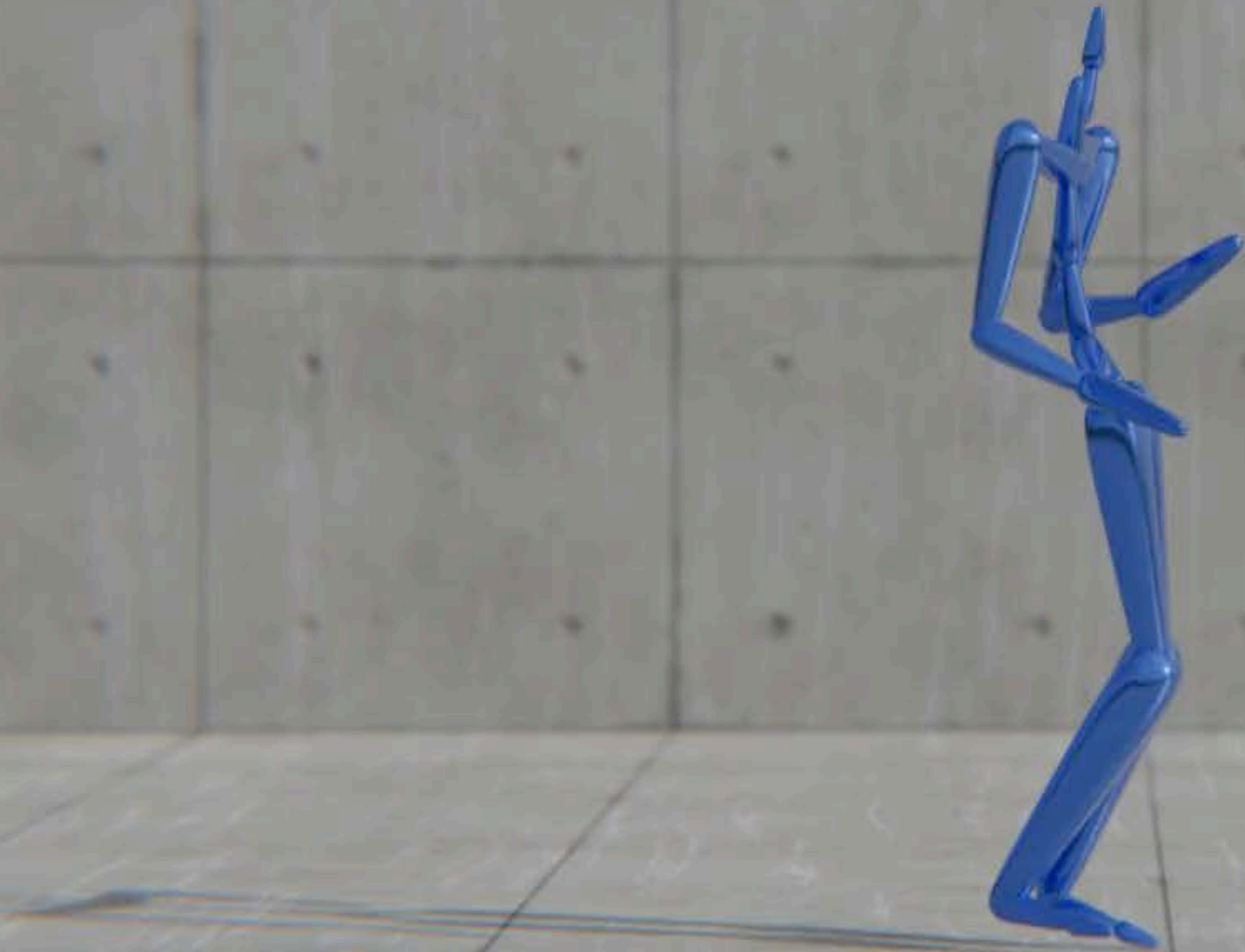
Co-Rotational



St. Venant-Kirchhoff



# Motion Capture Data Visualization



# 3DCG Software



# Blender



- Comprehensive 3DCG software
- **Free** and open source (GPL)
- Programmer-friendly (next slide)

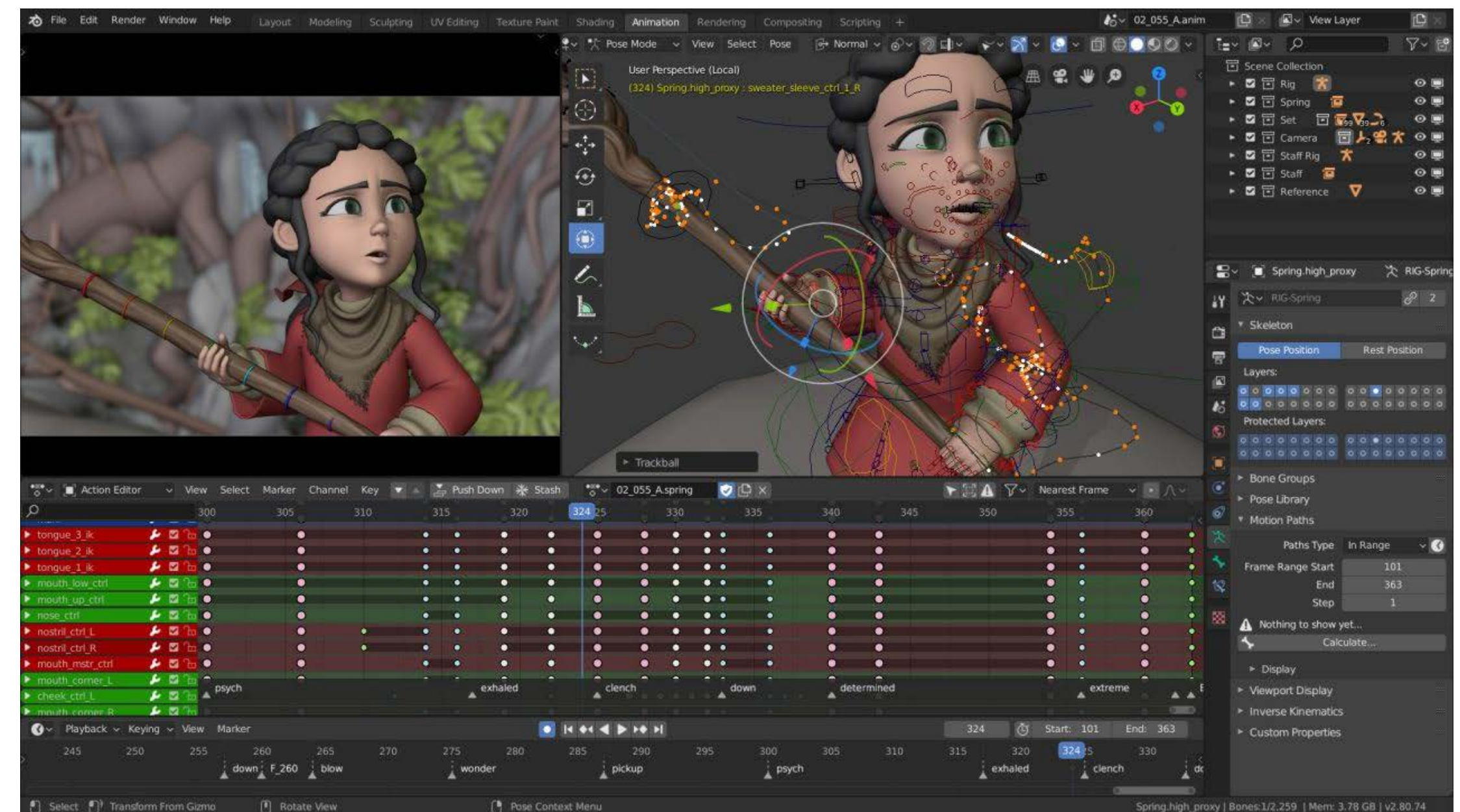


Image from <https://www.blender.org/>

# Blender is ... Programmer-Friendly! 🤝

- **Easy to install using command line**  
(see the right)
- Able to run on (display-less) computing servers (e.g., ABCI, Miyabi, AWS, etc.)
- Everything can be manipulated and automated via **Python APIs**
- **Programmer-friendly also means AI-friendly!!**

```
# Ubuntu
apt-get install blender

# macOS
brew install blender

# Windows
scoop install blender
```

# Other Software (Not Free)

- **3DCG animation and VFX**

- Maya
- 3ds Max
- Houdini
- ...

- **Game engine**

- Unreal Engine
- Unity
- ...

- **CAD**

- Autodesk Fusion
- Rhinoceros
- AutoCAD
- Solidworks
- ...

# 3DCG Tips for Researchers

How to Create Good Illustrations?



# Lighting Tips | Detailed in the following slides

- Use soft shadow (avoid hard shadow)
- Use cyclorama (infinity curve) background
- Environmental light is easy and useful



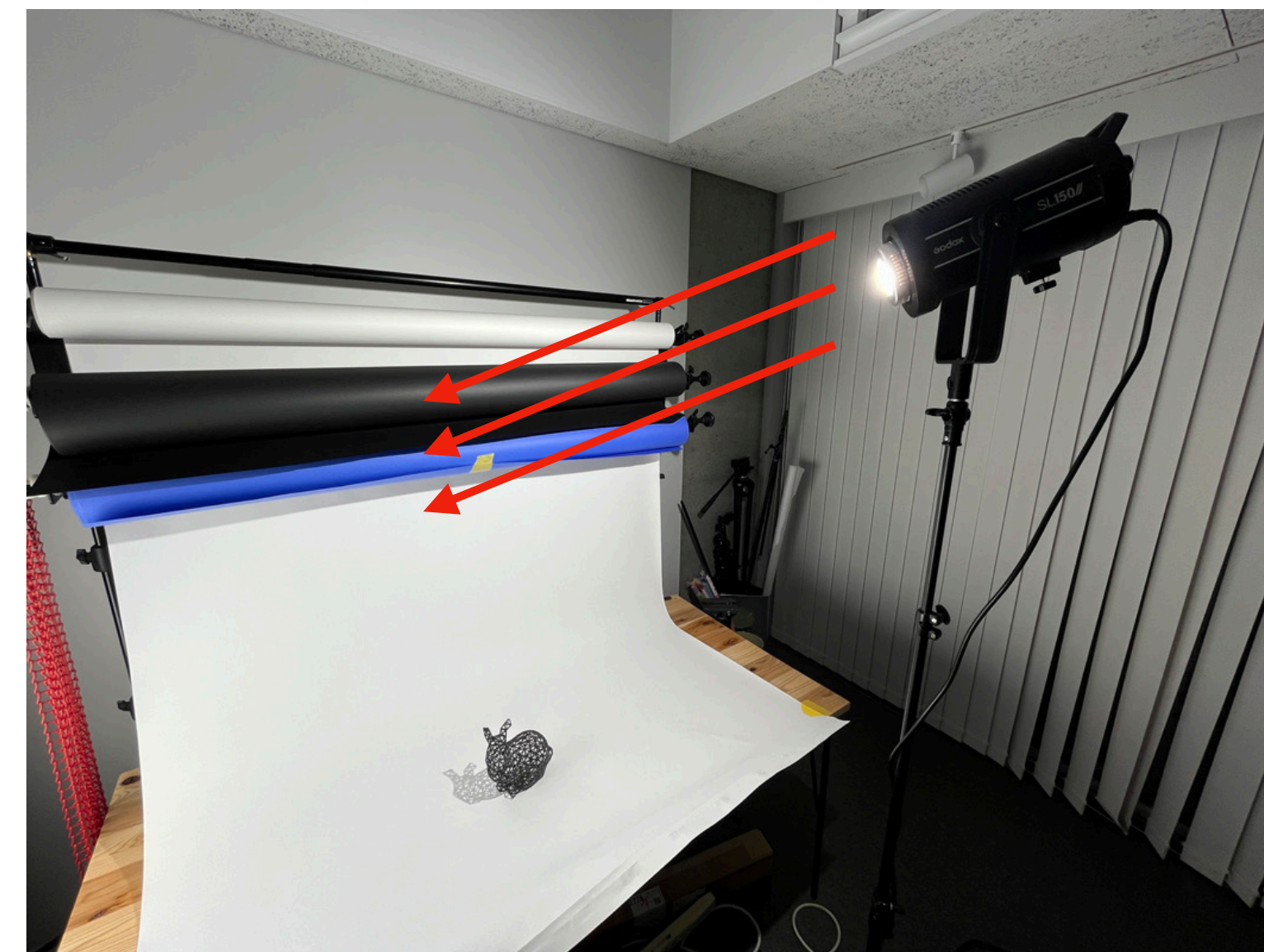
# Do you remember? — Week5 [Figures] by Koya Narumi

18

Avoid a non-diffused light



**Strong shadow → Bad**



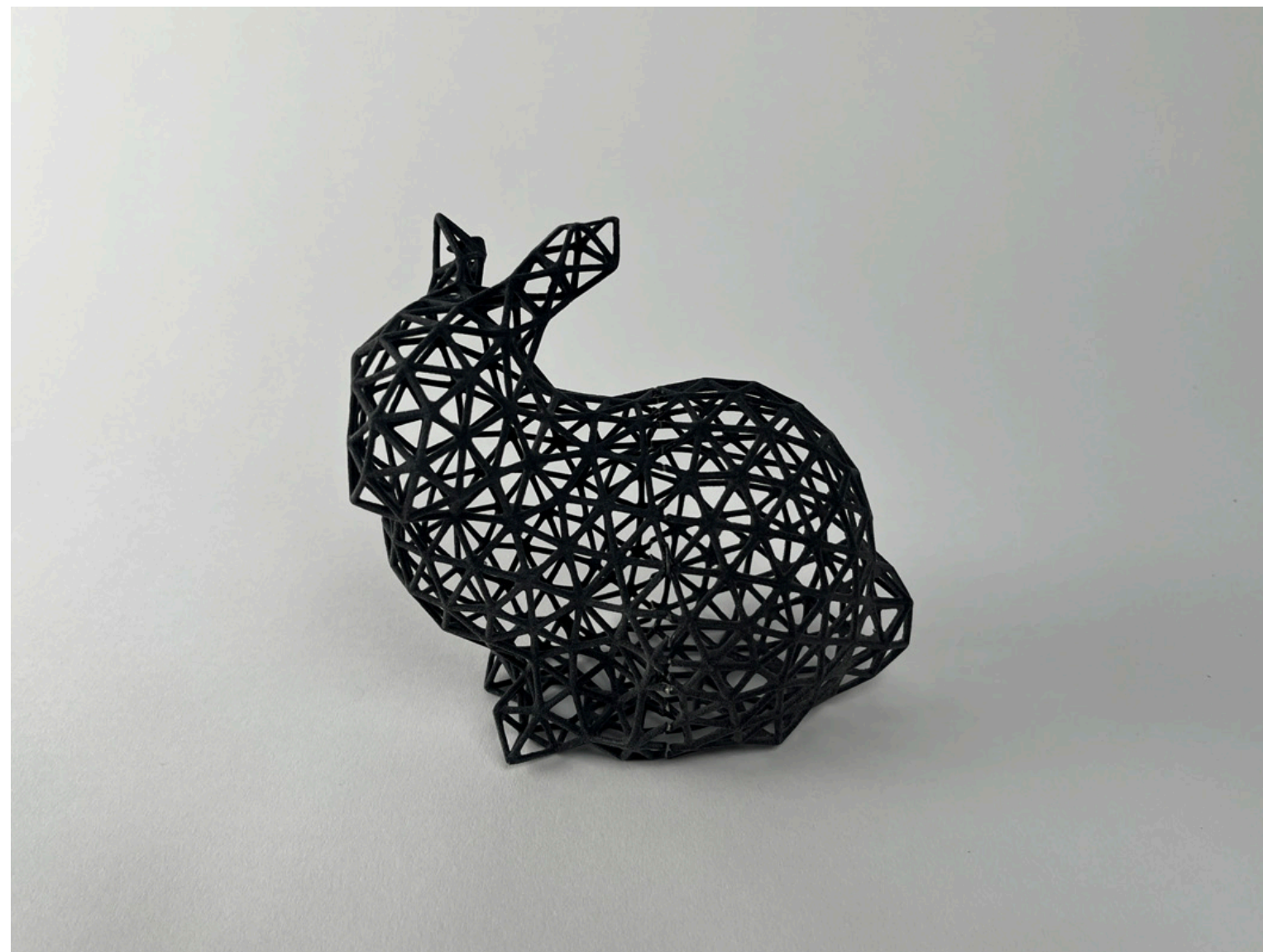
**Non-diffused light → Bad**

**Don't use a strong, non-diffused light** like a summer beach.

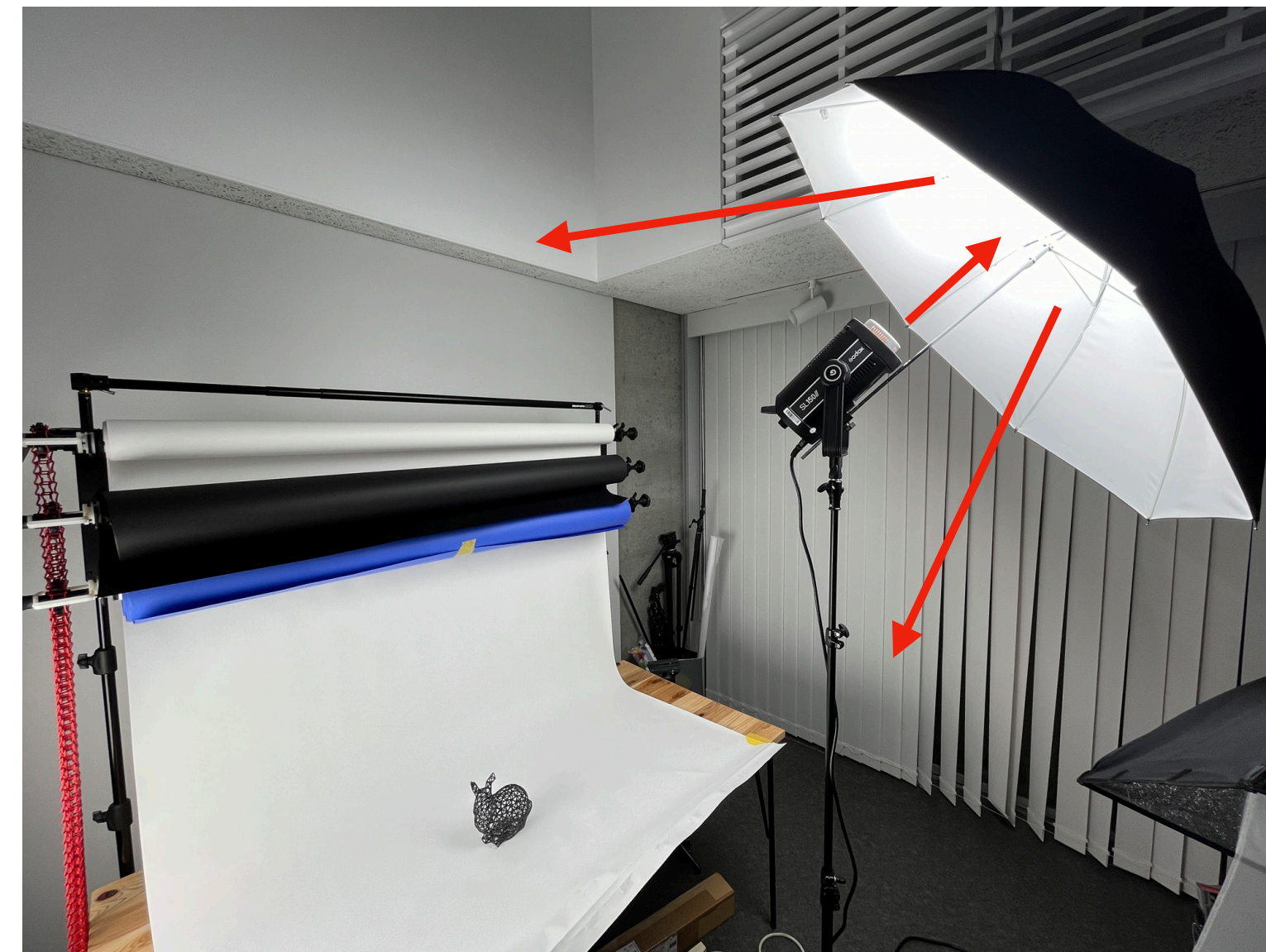
# Do you remember? — Week5 [Figures] by Koya Narumi

19

Use a diffused light



Soft shadow → Good

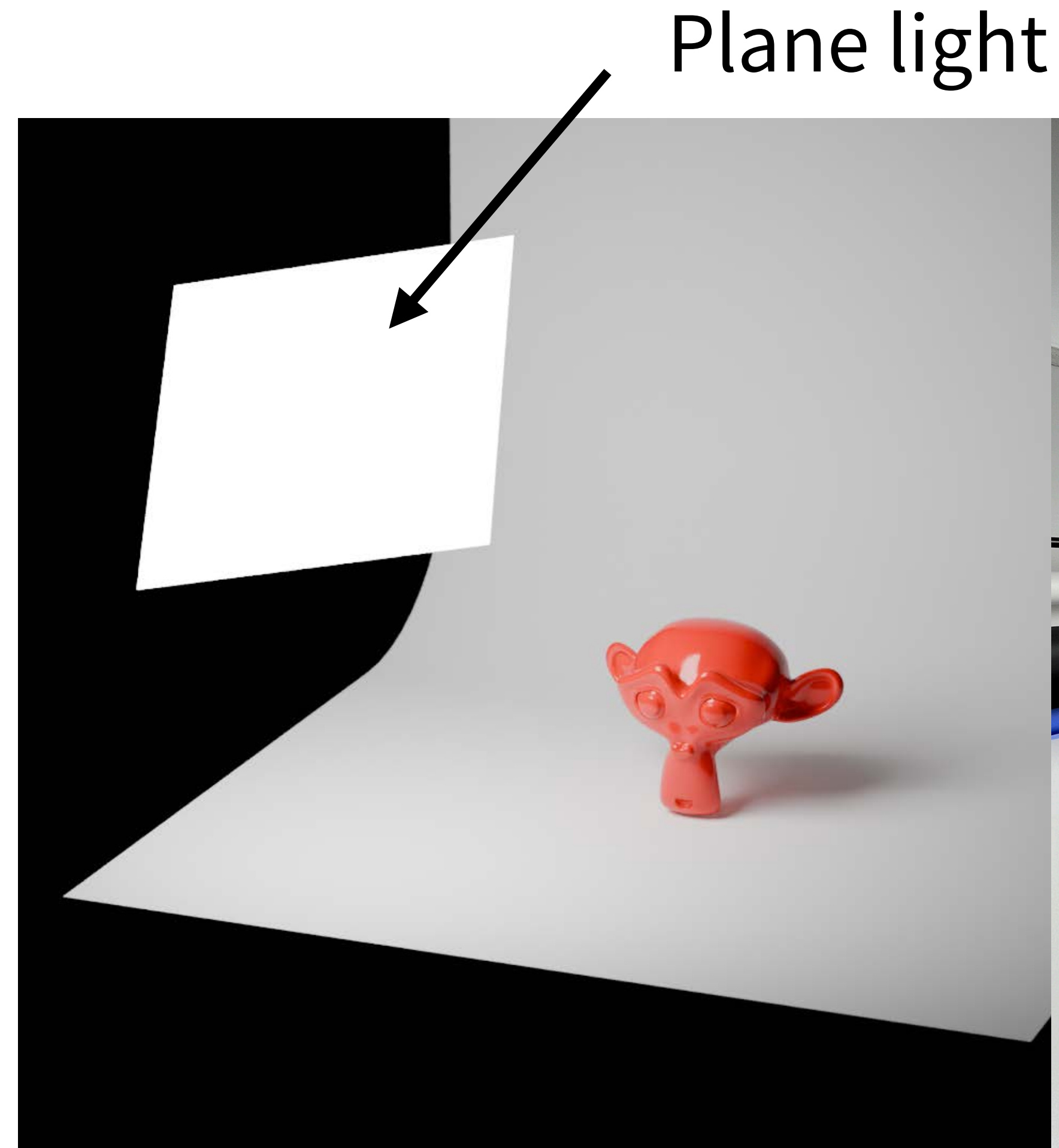


Diffused light → Good

**The diffused light** like a cloudy sky **is much better.**

# Lighting: Soft Shadow

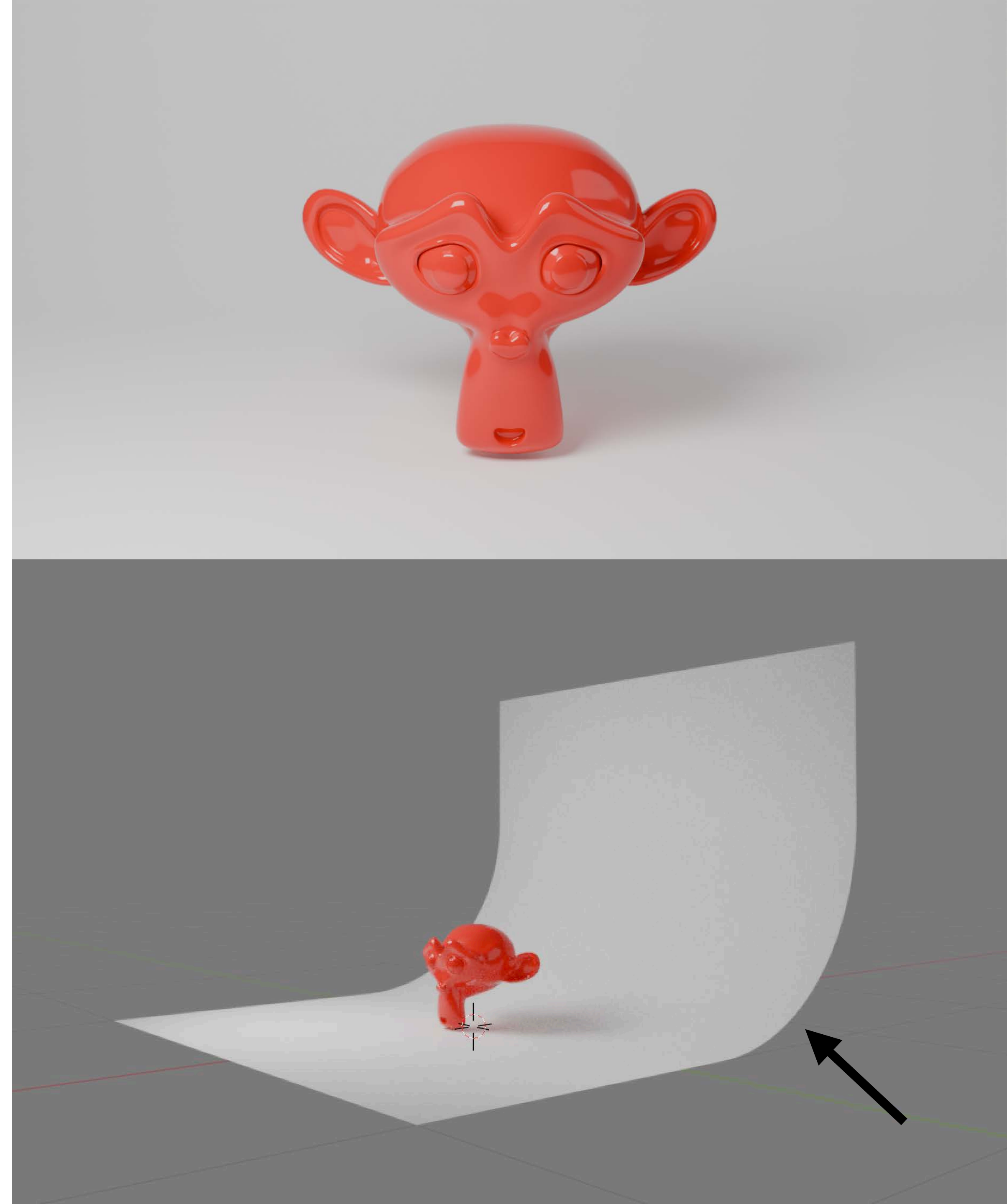
- 3DCG can easily create soft shadows by using **plane lights**
- No need of expensive equipments as in real-world photography 🤗



No need to buy this

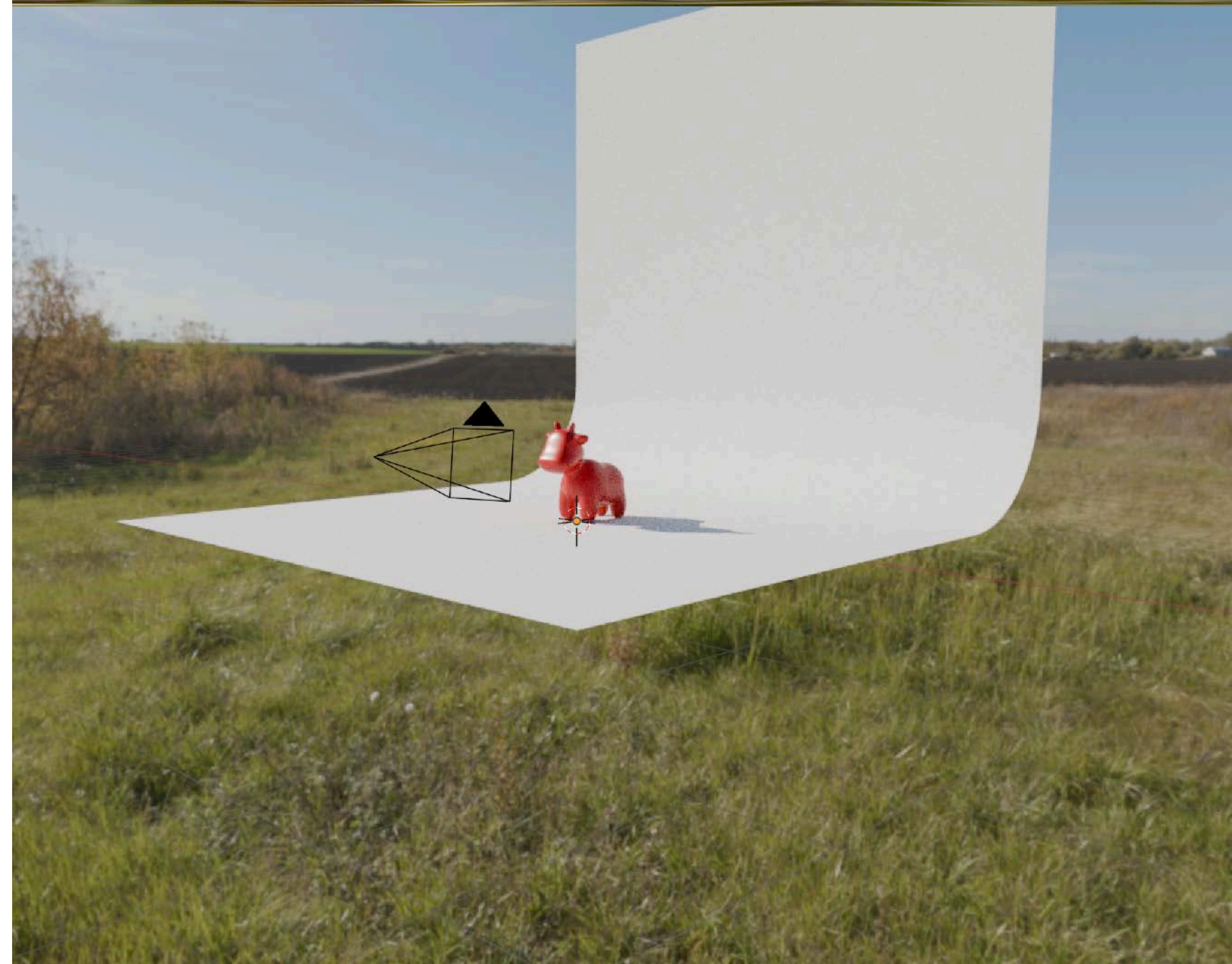
# Lighting: Cyclorama (Infinity Curve) Background

- Curved background to create **the illusion of an infinite space**
- Used in photography
- Also useful in 3DCG

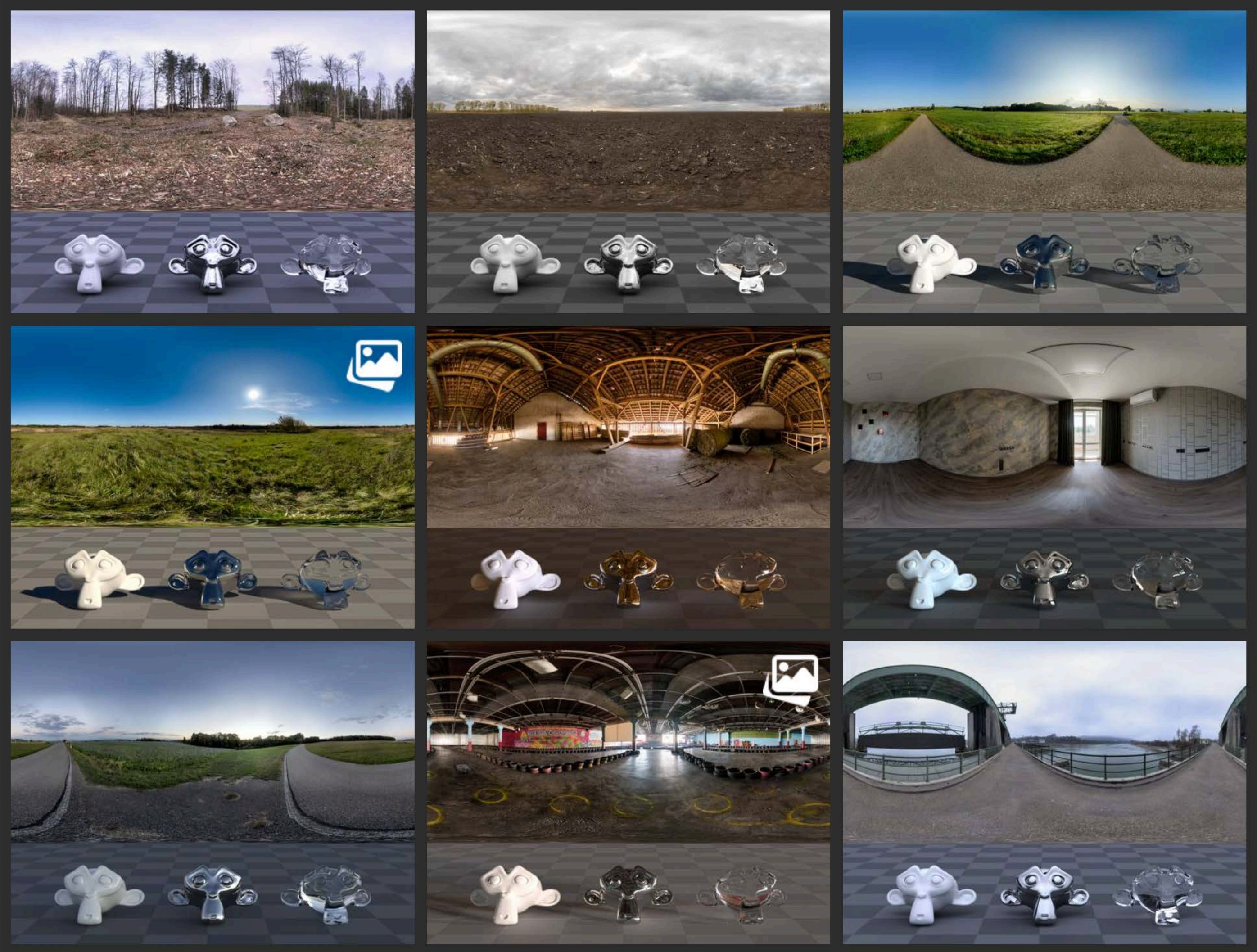


# Lighting: Environmental Map

- ... is a texture that represents the surrounding environment of a 3D scene
- Highly useful for creating realistic lighting (**very easy** 😊)



# Lighting: Where to Find HDR Images?



<https://hdri-haven.com/>

# Lighting: File Format for Environmental Maps

- **High Dynamic Range (HDR) image format (.hdr) is used**
- **Typical formats: unsigned int (8 bit/channel) — [0,255]**
  - 🤔 Cannot represent very bright ( $> 255$ ) light sources (overexposure)
- **HDR format: float (32 bit/channel) — [0,∞)**
  - 😊 Can represent more-than-1.0 values! 💪

# Materials Tips

- **Reflection** helps better shape perception 💡
- But be careful ⚠️  
Reflection increases **visual clutter**



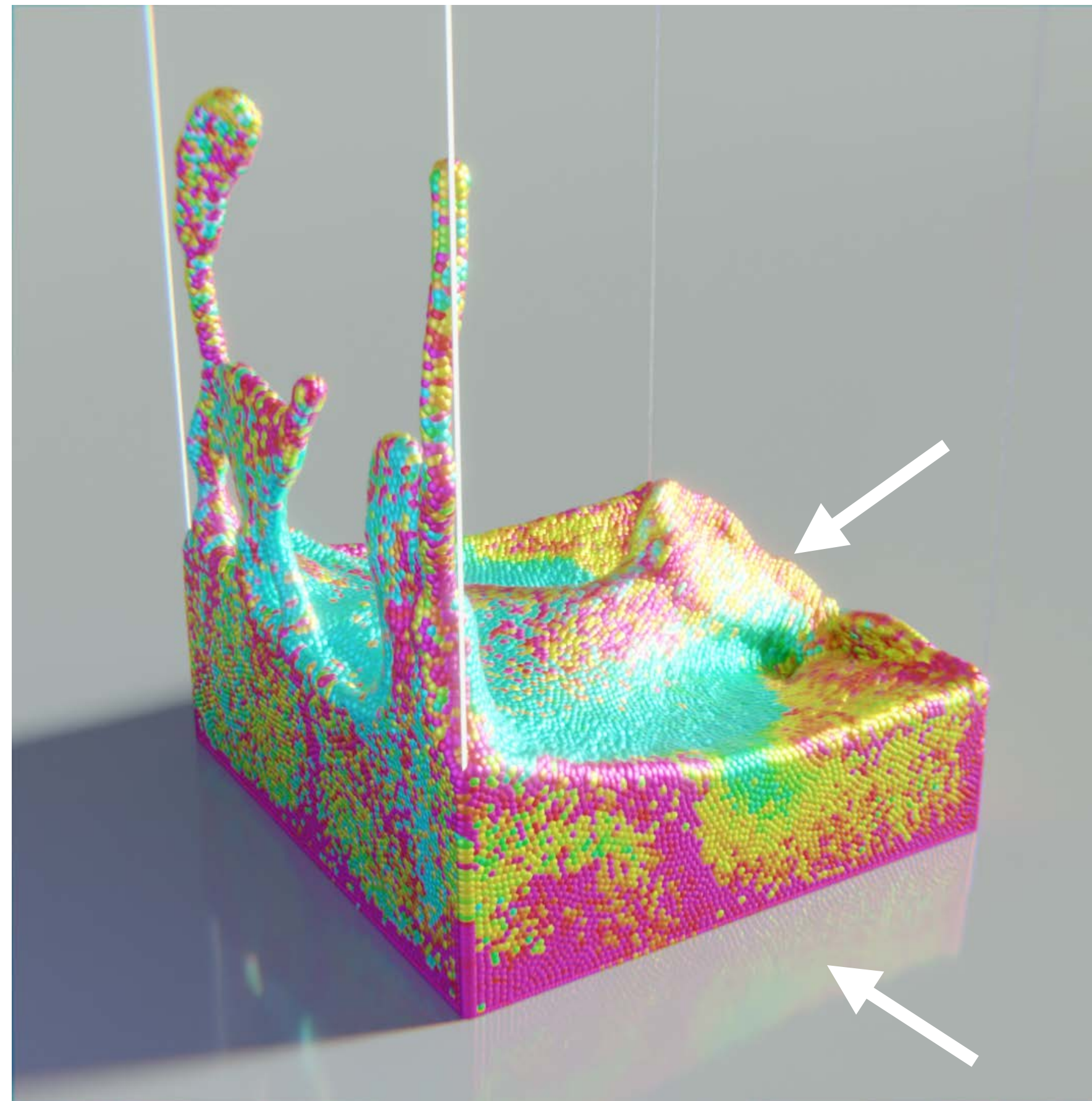
with reflection



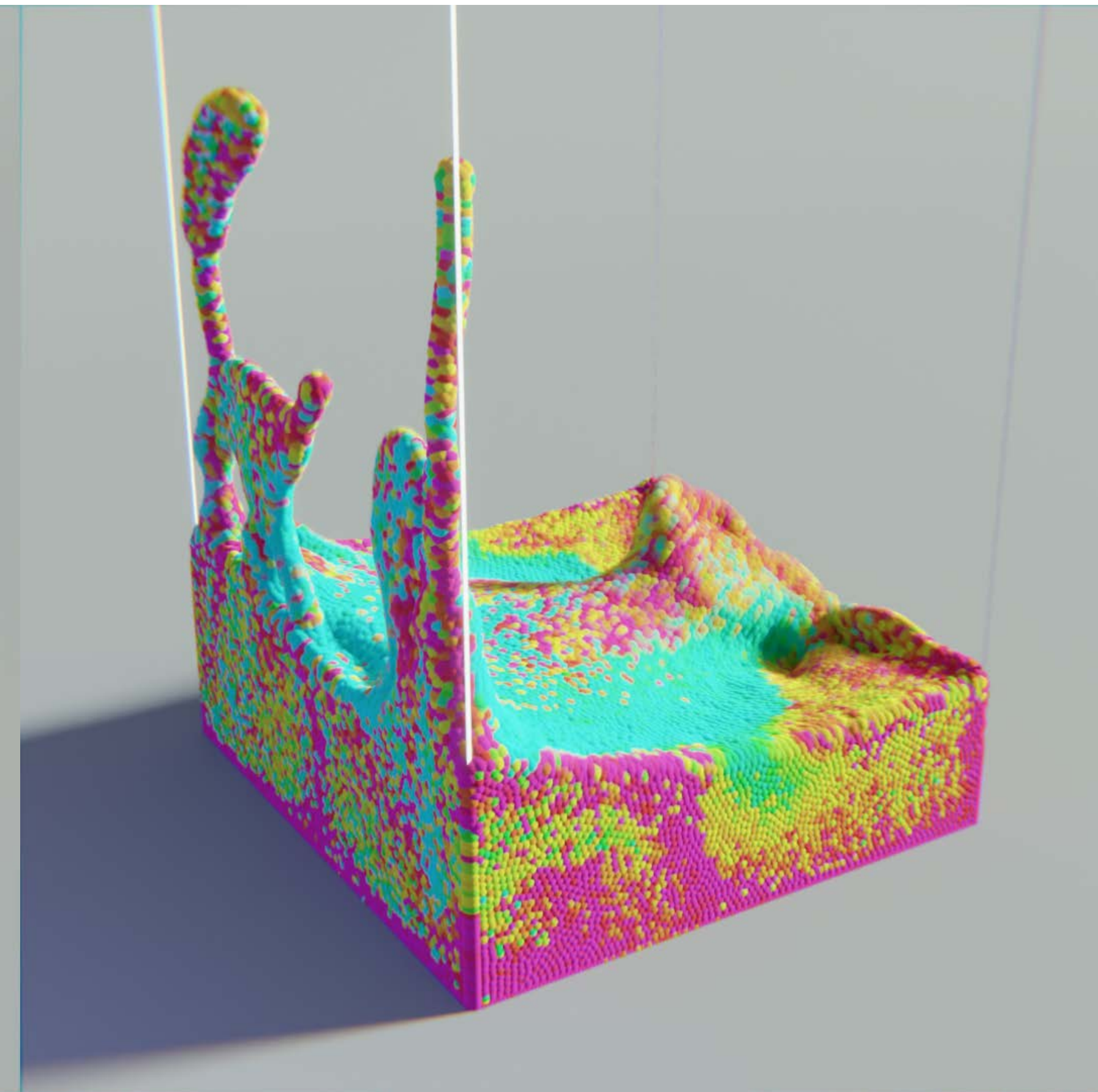
without reflection

# Materials Tips

- **Reflection** helps better shape perception 💡
- But be careful ⚠️  
Reflection increases **visual clutter**



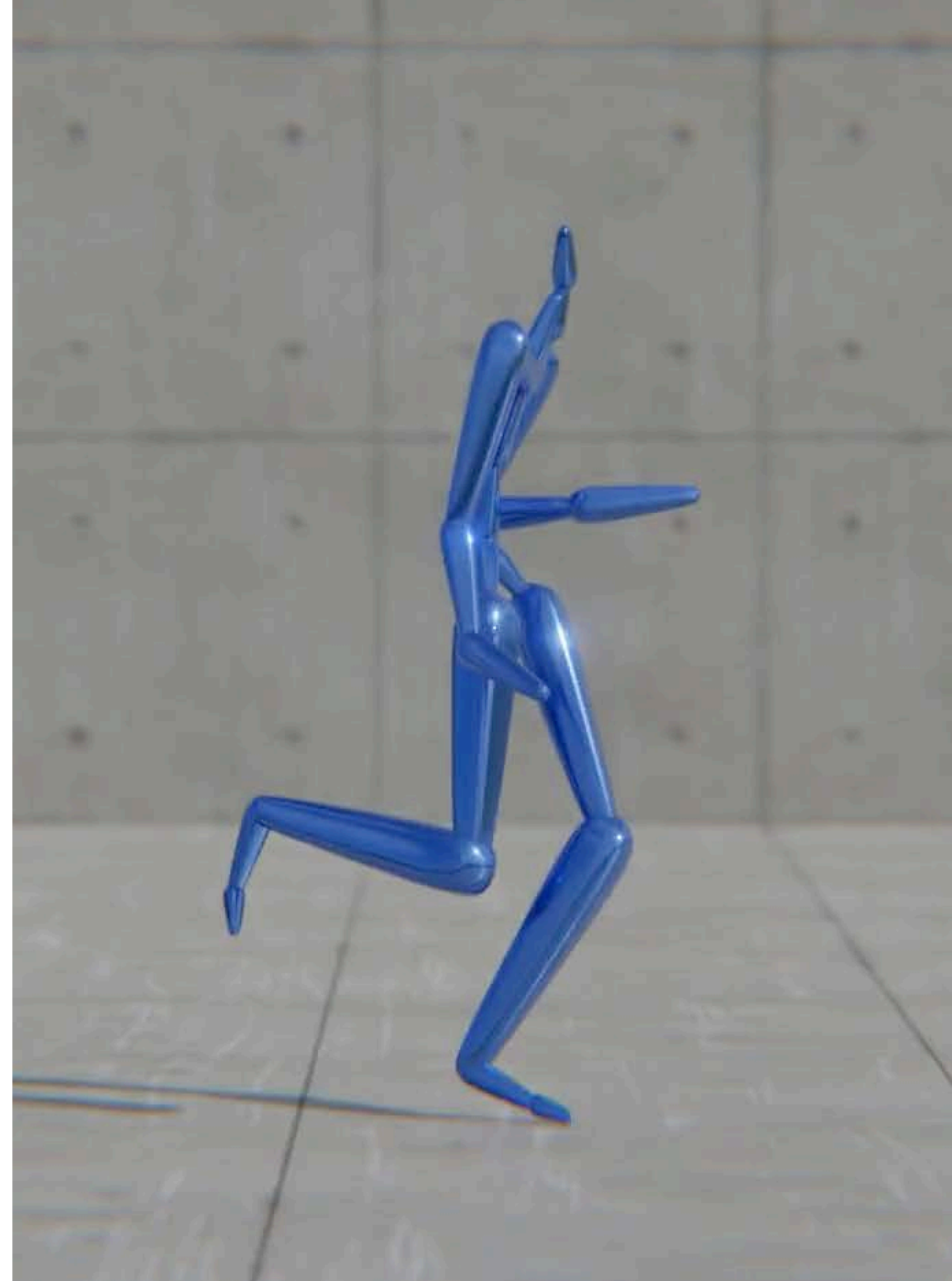
with reflection



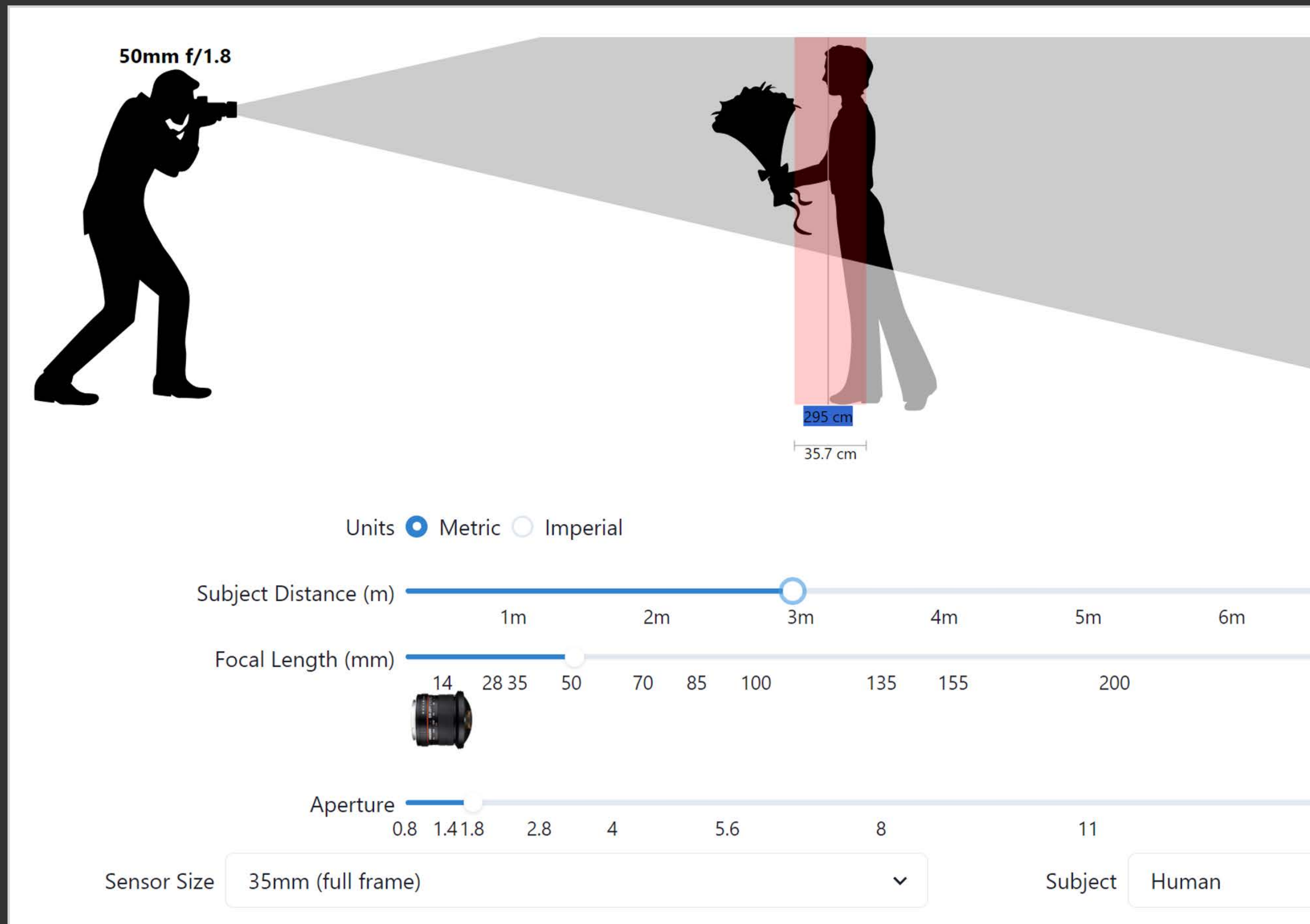
without reflection

# Camera Tips: Depth Blur

- Use **depth blur** for ...
  - Guiding viewers' attention 💡
  - Creating emotional expressions 💖
- But avoid using depth blur in most technical illustrations ⚠️



# Depth of Field (in real world)



<https://jherr.github.io/depth-of-field/>

# Scripting Tips

- **Everything that you can do with GUI can be done via Python API**
  - Automation is easy
  - Once scripting is done, it runs even on computing servers (e.g., ABCI)
- **It is easy to integrate Blender into your data processing pipeline**
  - E.g., Checkpoint visualization during the training of 3D-related ML models (3D shape generation, point cloud segmentation, human motion generation, etc.)
  - E.g., Visualization of fluid simulation results from various viewpoints (which is useful to check and validate the simulation behavior)

**Demonstration**



# Live Programming by Codex

*Knowing 3DCG tips helps you know what to ask AI*

# Codex wrote 200 lines

```
import os
import math
from math import radians
from mathutils import Vector

import bpy

ROOT = os.path.dirname(os.path.abspath(__file__))
START_PATH = os.path.join(ROOT, "start.blend")
HDR_PATH = os.path.join(ROOT, "tief_etz_4k.hdr")
OUT_DIR = os.path.join(ROOT, "out")
FRAME_DIR = os.path.join(OUT_DIR, "exploded_check")
OUT_PATH = os.path.join(FRAME_DIR, "frame_")

def look_at(obj, target):
    direction = Vector(target) - obj.location
    obj.rotation_euler = direction.to_track_quat("-Z", "Y").to_euler()

def scene_bounds():
    points = []
    for obj in bpy.context.scene.objects:
        if obj.type == "MESH":
            points.extend(obj.matrix_world @ Vector(corner) for corner in obj.bound_box)
    if not points:
        center = Vector((0, 0, 0))
        return center, 2.0, center, center

    min_corner = Vector((min(point[i] for point in points) for i in range(3)))
    max_corner = Vector((max(point[i] for point in points) for i in range(3)))
    center = (min_corner + max_corner) * 0.5
    radius = max((point - center).length for point in points)
    return center, max(radius, 1.0), min_corner, max_corner

def create_cyclorama(center, radius, floor_z, camera_location):
    forward = center - camera_location
    forward.z = 0
    if forward.length == 0:
        forward = Vector((0, 1, 0))
    forward.normalize()
    right = Vector((forward.y, -forward.x, 0))

    width = radius * 5.0
    front_depth = -radius * 3.0
    wall_depth = radius * 2.2
    curve_radius = radius * 0.9
    height = radius * 3.2
    x_left = -width * 0.5
    x_right = width * 0.5

    profile = [(front_depth, floor_z), (wall_depth - curve_radius, floor_z)]
    for index in range(1, 17):
        angle = -math.pi / 2 + (index / 16) * math.pi / 2
        y = wall_depth - curve_radius + curve_radius * math.cos(angle)
```

# Coding with Codex, 20 min

```
import os
import math
from math import radians
from mathutils import Vector

import bpy

ROOT = os.path.dirname(os.path.abspath(__file__))
START_PATH = os.path.join(ROOT, "start.blend")
HDR_PATH = os.path.join(ROOT, "tief_etz_4k.hdr")
OUT_DIR = os.path.join(ROOT, "out")
FRAME_DIR = os.path.join(OUT_DIR, "exploded_check")
OUT_PATH = os.path.join(FRAME_DIR, "frame_")

def look_at(obj, target):
    direction = Vector(target) - obj.location
    obj.rotation_euler = direction.to_track_quat("-Z", "Y").to_euler()

def scene_bounds():
    points = []
    for obj in bpy.context.scene.objects:
        if obj.type == "MESH":
            points.extend(obj.matrix_world @ Vector(corner) for corner in obj.bound_box)
    if not points:
        center = Vector((0, 0, 0))
        return center, 2.0, center, center

    min_corner = Vector((min(point[i] for point in points) for i in range(3)))
    max_corner = Vector((max(point[i] for point in points) for i in range(3)))
    center = (min_corner + max_corner) * 0.5
    radius = max((point - center).length for point in points)
    return center, max(radius, 1.0), min_corner, max_corner

def create_cyclorama(center, radius, floor_z, camera_location):
    forward = center - camera_location
    forward.z = 0
    if forward.length == 0:
        forward = Vector((0, 1, 0))
    forward.normalize()
    right = Vector((forward.y, -forward.x, 0))

    width = radius * 5.0
    front_depth = -radius * 3.0
    wall_depth = radius * 2.2
    curve_radius = radius * 0.9
    height = radius * 3.2
    x_left = -width * 0.5
    x_right = width * 0.5

    profile = [(front_depth, floor_z), (wall_depth - curve_radius, floor_z)]
    for index in range(1, 17):
        angle = -math.pi / 2 + (index / 16) * math.pi / 2
        y = wall_depth - curve_radius + curve_radius * math.cos(angle)
        z = floor_z + curve_radius + curve_radius * math.sin(angle)
        profile.append((y, z))
    profile.append((wall_depth, floor_z + height))

    vertices = []
    for depth_z in profile:
        left = center + forward * depth_z + right * x_left
        right_point = center + forward * depth_z + right * x_right
        vertices.append((left.x, left.y, z))
        vertices.append((right_point.x, right_point.y, z))

    faces = []
    for index in range(len(profile) - 1):
        left_a = index + 2
        right_a = left_a + 1
        left_b = left_a + 2
        right_b = left_b + 2
        faces.append((left_a, right_a, right_b, left_b))

    mesh = bpy.data.meshes.new("cyclorama_mesh")
    mesh.from_pydata(vertices, [], faces)
    mesh.update()

    cyclorama = bpy.data.objects.new("Dark Gray Cyclorama", mesh)
    bpy.context.collection.objects.link(cyclorama)

    material = bpy.data.materials.new("dark_gray_cyclorama")
    material.use_nodes = True
    bsdf = material.node_tree.nodes["Principled BSDF"]
    bsdf.inputs["Base Color"].default_value = (0.08, 0.08, 0.08, 1)
    bsdf.inputs["Roughness"].default_value = 0.75
    cyclorama.data.materials.append(material)
    return cyclorama

def object_center(obj):
    points = [obj.matrix_world @ Vector(corner) for corner in obj.bound_box]
    return sum(points, Vector()) / len(points)

def explosion_offset(obj, center, radius):
    name = obj.name.lower()
    amount = radius * 0.35

    if "top" in name:
        return Vector((0, 0, amount))
    if "bottom" in name:
        return Vector((0, 0, -amount))
    if "display" in name:
        return Vector((0, 0, -amount))
    if "right" in name:
        return Vector((0, 0, -amount))
    if "left" in name:
        return Vector((0, 0, amount))
    if "button" in name:
        return Vector((0, -amount * 0.55, amount * 0.35))

    direction = object_center(obj) - center
    direction.z = 0.35
    if direction.length == 0:
        return Vector((0, 0, 0))
    direction.normalize()
    return direction * amount * 0.35

def animate_exploded_view(center, radius, start_frame, hold_frame, end_frame):
    for obj in bpy.context.scene.objects:
        if obj.type != "MESH":
            continue
        if "cyclorama" in obj.name.lower():
            continue
        original_location = obj.location.copy()
        exploded_location = original_location + explosion_offset(obj, center, radius)

        obj.location = original_location
        obj.keyframe_insert(data_path="location", frame=start_frame)
        obj.keyframe_insert(data_path="location", frame=hold_frame)
        obj.location = exploded_location
        obj.keyframe_insert(data_path="location", frame=end_frame)

os.makedirs(OUT_DIR, exist_ok=True)
camera_location = object_center(cyclorama)
camera_location = Vector((radius * 1.2, -radius * 1.8, radius * 0.75))

bpy.ops.object.camera_add(location=camera_location)
camera = bpy.context.object
camera.data.lens = 28
camera.data.clip_end = radius * 28
look_at(camera, center)
bpy.context.scene.camera = camera

create_cyclorama(center, radius, min_corner.z - radius * 0.18, camera_location)

pivot = bpy.data.objects.new("Camera Orbit Pivot", None)
pivot.empty_display_type = "PLAIN_AXES"
pivot.location = center
bpy.context.collection.objects.link(pivot)

camera.parent = pivot
camera.matrix_parent_inverse = pivot.matrix_world.inverted()
track = camera.constraints.new(type="TRACK_1D")
track.target = pivot
track.track_axis = "TRACK_NEGATIVE_Z"
track.up_axis = "UP_Y"

scene = bpy.context.scene
scene.frame_start = 1
scene.frame_end = 24
scene.render.fps = 24
animate_exploded_view(center, radius, scene.frame_start, 5, scene.frame_end)

pivot.rotation_euler = (0, 0, radians(-7.5))
pivot.keyframe_insert(data_path="rotation_euler", frame=scene.frame_start)
pivot.rotation_euler = (0, 0, radians(7.5))
pivot.keyframe_insert(data_path="rotation_euler", frame=scene.frame_end)

world = bpy.context.scene.world or bpy.data.worlds.new("World")
bpy.context.scene.world = world
world.use_nodes = True
nodes = world.node_tree.nodes
[links.unlink(world_tree_links) for links in world_tree_links]
nodes.clear()

env = nodes.new("ShaderNodeTexEnvironment")
env.inputs["Environment"] = bpy.data.images.load(HDR_PATH)
background = nodes.new("ShaderNodeBackground")
background.inputs["Color"] = (0, 0, 0, 1)
output = nodes.new("ShaderNodeOutputShader")
[links.new(env.outputs["Color"], background.inputs["Color"])]
[links.new(background.outputs["Background"], output.inputs["Surface"])]

scene.render.engine = "CYCLES"
scene.cycles.samples = 8
scene.cycles.use_denoising = True
scene.cycles.use_fast_gi = True
scene.render.resolution_x = 488
scene.render.resolution_y = 368
scene.render.use_settings_file_format = "PNG"
scene.render.filepath = OUT_PATH

bpy.ops.render.render(animation=True)
print(f"Rendered animation frames to {FRAME_DIR}")
```



# C.f.) Manual setup, 20 min



The 3D model was provided by George Kayesi for tutorial purposes  
<https://www.youtube.com/watch?v=IC0-9b0Rv2g>

# Mastering 3DCG: Where to Start?

# Resources

## The "Donut" Tutorial

<https://www.youtube.com/@blenderguru>



**Blender Donut Tutorial (2025)**

Blender Guru · コース  
8本の動画 更新: 7日前

再生 コメント

Learn essential Blender skills in this free course.  
This version uses Blender 5.0

- 1 **Blender Donut Tutorial Part 1 (2026)** 28:30
- 2 **Blender Basic Modelling Tutorial (Donut Part 2)** 30:51
- 3 **Blender Basic Organic Modelling Tutorial (Donut Part 3)** 31:23
- 4 **Blender Basic Materials Tutorial (Donut Part 4)** 33:33

# Resources

- Exploded View Animation of Objects in Blender (Tutorial)
  - <https://www.youtube.com/watch?v=IC0-9b0Rv2g>
- Illustrating Geometry (Keenan Crane)
  - <https://www.cs.cmu.edu/~kmcrane/Projects/Other/IllustratingGeometry.pdf>
- Multi-scale Multi-physics Heart Simulator UT-Heart
  - <https://www.youtube.com/watch?v=2LPboySOSvo>
- Blender Python scripting examples
  - <https://github.com/yuki-koyama/blender-cli-rendering>

# Summary



In general, **good illustrations** ...

1. help **audience better understand** your research 💡 , and
2. increase presentation appeal 🌟 (→ **buzz** 🚀).

**3DCG** is (sometimes) an effective technique for these purposes.