```
# create_and_export_ring.py

# Usage:

# - Save this file in a folder.

# - Open Blender and run from Scripting tab OR run from terminal:

# blender --background --python create_and_export_ring.py

#

# Output: ring.glb (saved next to this script)



import bpy, os

from math import pi




# --- helper to clear scene ---

def clear_scene():

  bpy.ops.object.select_all(action='SELECT')

  bpy.ops.object.delete(use_global=False)
  # remove unused data blocks
  for block in bpy.data.meshes:
      if block.users == 0:
          bpy.data.meshes.remove(block)
  for block in bpy.data.materials:
      if block.users == 0:
          bpy.data.materials.remove(block)

# --- create ring, gem, prongs, materials, lights, camera, animation ---
def build_scene():
  # ring group (parent)
  grp = bpy.data.objects.new("Ring_Group", None)
  bpy.context.collection.objects.link(grp)


  # --- Ring (torus) ---
  bpy.ops.mesh.primitive_torus_add(major_radius=1.35, minor_radius=0.18, major_segments=96, minor_segments=48, location=(0,0,0))

  ring = bpy.context.active_object

  ring.name = "Silver_Ring"

  ring.rotation_euler[0] = pi/2 # lay flat

  ring.parent = grp



  # Silver material (PBR)

  silver = bpy.data.materials.new("Silver_PBR")

  silver.use_nodes = True

  nodes = silver.node_tree.nodes

  links = silver.node_tree.links

  nodes.clear()

  # Principled BSDF setup

  out = nodes.new(type="ShaderNodeOutputMaterial")

  principled = nodes.new(type="ShaderNodeBsdfPrincipled")
  principled.inputs["Metallic"].default_value = 1.0
  principled.inputs["Roughness"].default_value = 0.06
  principled.inputs["Specular"].default_value = 0.6
  principled.location = (-200, 0)
  out.location = (100, 0)
  links.new(principled.outputs["BSDF"], out.inputs["Surface"])
  ring.data.materials.append(silver)


  # --- Gem (red coral) ---
  bpy.ops.mesh.primitive_uv_sphere_add(radius=0.45, segments=64, ring_count=64, location=(0,0,0.45))
  gem = bpy.context.active_object
  gem.name = "Red_Coral"
  # scale to oval
  gem.scale[1] = 1.35
  gem.parent = grp
```

```
# Gem material
gem_mat = bpy.data.materials.new("Coral_PBR")
gem_mat.use_nodes = True
g_nodes = gem_mat.node_tree.nodes
g_links = gem_mat.node_tree.links
g_nodes.clear()
gout = g_nodes.new(type="ShaderNodeOutputMaterial")
gpr = g_nodes.new(type="ShaderNodeBsdfPrincipled")
gpr.inputs["Base Color"].default_value = (1.0, 0.12, 0.12, 1.0)
gpr.inputs["Roughness"].default_value = 0.15
gpr.inputs["Specular"].default_value = 0.8
gpr.inputs["Clearcoat"].default_value = 0.9
gpr.inputs["Clearcoat Roughness"].default_value = 0.02
g_links.new(gpr.outputs["BSDF"], gout.inputs["Surface"])
```

```
gem.data.materials.append(gem_mat)


# --- Prongs (4) ---
prong_mat = silver # use same silver material
import math
for i in range(4):
    angle = i * (math.pi/2)
    # create a prong as a box then shape it a bit
    bpy.ops.mesh.primitive_cube_add(size=1, location=(0,0,0.18))
    p = bpy.context.active_object
    p.name = f"Prong_{i+1}"
    p.scale[0] = 0.04
    p.scale[1] = 0.28
    p.scale[2] = 0.12
    # position around gem
    radius = 0.7
    px = radius * math.cos(angle)
    py = radius * math.sin(angle)
    p.location = (px, py, 0.18)
    p.rotation_euler[2] = angle + 0.35
    p.parent = grp
    p.data.materials.append(prong_mat)
    # add slight bevel modifier for nicer look
    mod = p.modifiers.new(name="Bevel", type='BEVEL')
    mod.width = 0.008
    mod.segments = 3


# --- Floor (for nicer export previews) ---
bpy.ops.mesh.primitive_plane_add(size=10, location=(0,0,-0.6))
floor = bpy.context.active_object
```

```
floor.name = "Studio_Floor"
floor_mat = bpy.data.materials.new("Floor_MAT")
floor_mat.use_nodes = True
fm_nodes = floor_mat.node_tree.nodes
fm_nodes["Principled BSDF"].inputs["Base Color"].default_value = (0.03, 0.03, 0.03, 1)
fm_nodes["Principled BSDF"].inputs["Roughness"].default_value = 0.75
floor.data.materials.append(floor_mat)
floor.parent = grp

# --- Lights (attractive studio lights) ---
# Key area light (Rect light)
bpy.ops.object.light_add(type='AREA', location=(3.2, -2.8, 3.5), rotation=(0.8, 0, 0.8))
key = bpy.context.active_object
key.name = "Key_Area"
key.data.energy = 800
key.data.size = 2.5
# Fill
bpy.ops.object.light_add(type='AREA', location=(-3.0, 2.8, 1.8), rotation=(1.1, 0, -0.6))
fill = bpy.context.active_object
fill.name = "Fill_Area"
fill.data.energy = 250
fill.data.size = 2.2
# Rim
bpy.ops.object.light_add(type='AREA', location=(0, -5.2, 4.4), rotation=(1.2, 0, 0))
rim = bpy.context.active_object
rim.name = "Rim_Area"
rim.data.energy = 420
rim.data.size = 2.0
# small warm spot for sparkle
bpy.ops.object.light_add(type='SPOT', location=(2.2, -1.5, 4.0))
```

```
spot = bpy.context.active_object

spot.data.energy = 1200

spot.data.spot_size = 0.6

spot.data.shadow_soft_size = 0.12

spot.name = "Spark_Spot"



# Parent lights to group? No, keep them in scene.

# --- Camera ---

bpy.ops.object.camera_add(location=(4.6, -4.2, 2.8), rotation=(1.08, 0, 0.78))

cam = bpy.context.active_object

cam.name = "RenderCam"

bpy.context.scene.camera = cam



# --- World (simple HDR-like using nodes) ---

world = bpy.context.scene.world
```

```
world.use_nodes = True
wn = world.node_tree.nodes
wl = world.node_tree.links
wn.clear()
output = wn.new(type='ShaderNodeOutputWorld')
background = wn.new(type='ShaderNodeBackground')
# subtle bluish studio environment
background.inputs['Color'].default_value = (0.03, 0.03, 0.035, 1)
background.inputs['Strength'].default_value = 0.9
wl.new(background.outputs['Background'], output.inputs['Surface'])

# --- Animation: rotate the group for 360° over frame range ---
bpy.context.scene.frame_start = 1
bpy.context.scene.frame_end = 250
grp.rotation_euler = (0, 0, 0)
grp.keyframe_insert(data_path="rotation_euler", frame=1)
grp.rotation_euler = (0, 0, 2*pi)
grp.keyframe_insert(data_path="rotation_euler", frame=250)

# Set interpolation to linear for constant rotation speed
for fcurve in grp.animation_data.action.fcurves:
    for kf in fcurve.keyframe_points:
        kf.interpolation = 'LINEAR'

# make sure objects cast/receive shadows
for obj in [ring, gem] + [o for o in grp.children if "Prong" in o.name] + [floor]:
    obj.cycles_visibility.shadow = True
    if hasattr(obj, "cycles"):
        pass
```

```
  return grp


# --- export glb ---
def export_glb(output_path):
  # ensure export addons enabled (glTF exporter included in Blender by default)
  bpy.ops.export_scene.gltf(
      filepath=output_path,
      export_format='GLB',
      export_texture_transform=True,
      export_cameras=True,
      export_lights=True,
      export_extras=False,
      export_yup=True,
      export_apply=True,
      export_animations=True,
      export_materials='EXPORT'
  )


if __name__ == "__main__":
  # determine script folder & set output path
  script_file = os.path.realpath(__file__)
  script_dir = os.path.dirname(script_file)
  out_file = os.path.join(script_dir, "ring.glb")

  clear_scene()
  build_scene()

  print("Exporting to:", out_file)
  export_glb(out_file)
  print("Done. Exported ring.glb")
```