# What Is Isaac Sim?

**在 NVIDIA Omniverse 平台上的有多種參考應用程式可建構，如:**

- **NVIDIA Isaac Sim™:** 預先安裝了所有必要的組件
- **USD Explorer:** 自定義安裝所需功能與介面樣式

**Isaac Sim → 實體機器人與模擬環境間的直接通訊:**

**ROS 2(Robot Operating System):** Isaac Sim 為 ROS 2 提供橋接 API，用於實機與模擬之間的通訊

**NVIDIA Isaac ROS:** 一套高效能、硬體加速的 ROS 2 套件集合，用於打造自主機器人(autonomous robots)系統。

**Isaac Sim 有三種操作模式可選擇，它們執行相同的操作並達到相同的結果:**

1. GUI
2. Extensions
3. Standalone Python

# GUI

# Isaac Sim 基本使用 - GUI

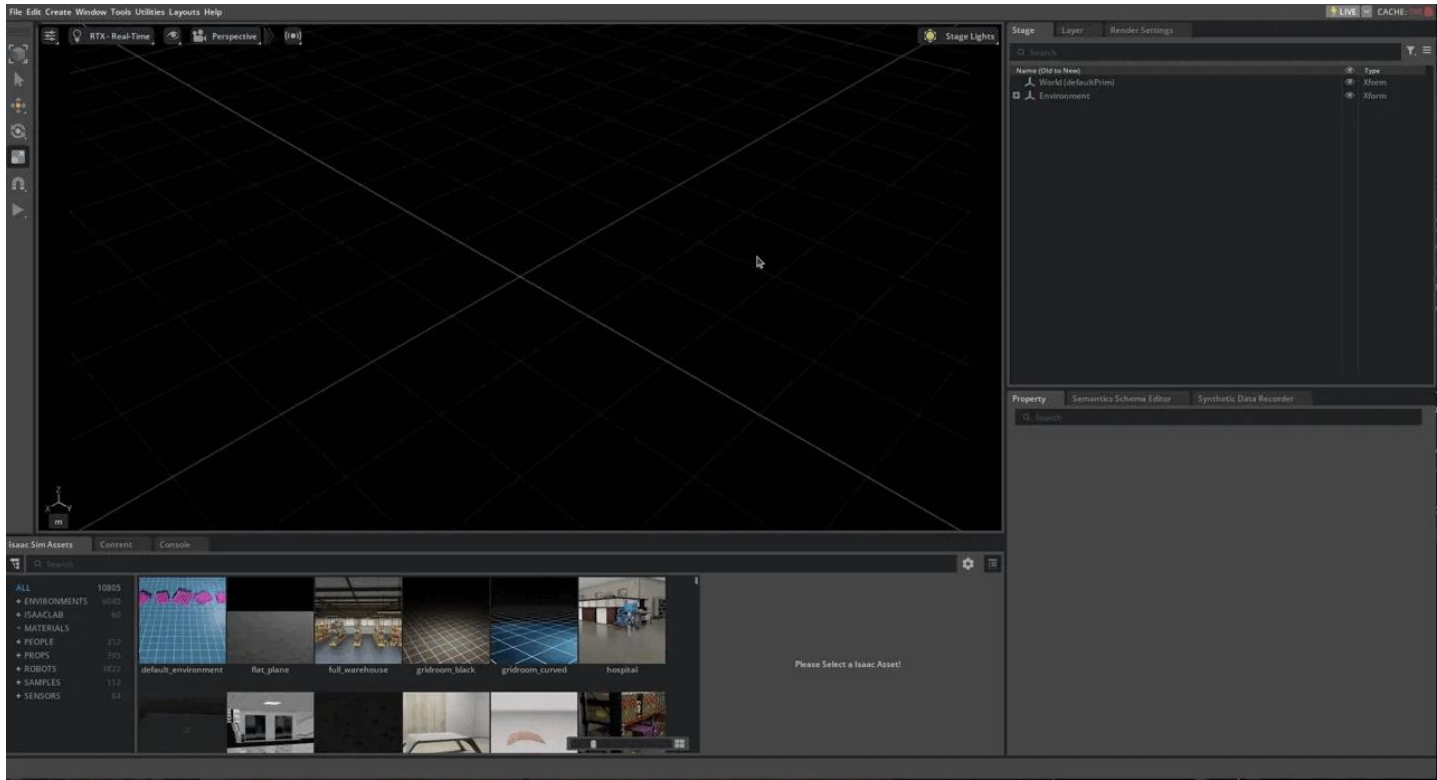
**啟動:** 從安裝根目錄執行「isaac-sim.selector」以啟動 Isaac Sim。

1. **新增場景：** File → New.
2. **增加地平面:** Create → Physics → Ground Plane
3. **增加光源:** Create → Lights → Distant Light (可選不同光源)

- **注意: 如果場景中存在光源，但沒有反射光線的物體，場景仍然會很暗。**

4. **加入「可視化」立方體:** Create → Shape → Cube

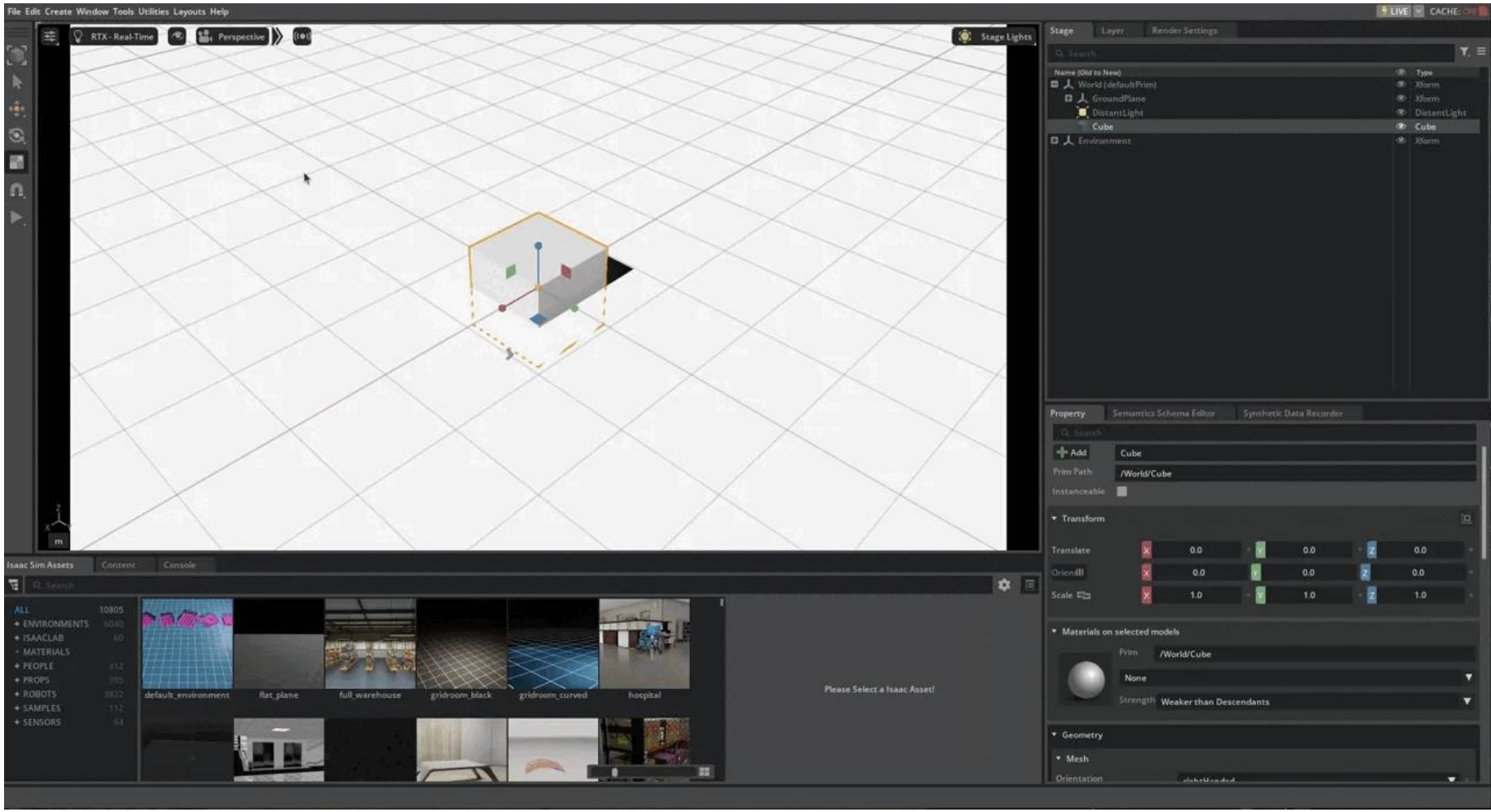
- **「可視化」立方體是指不附加任何物理屬性的立方體，例如，沒有質量、沒有碰撞。這種立方體不會因重力作用而下落，也不會與其他物體碰撞。**



# 移動、旋轉與縮放 - GUI

- **控制工具**（Gizmo）：左側工具列選擇或使用快捷鍵
  - **W**：Move（移動）
  - **E**：Rotate（旋轉）
  - **R**：Scale（縮放）
  - **Esc**：取消選取

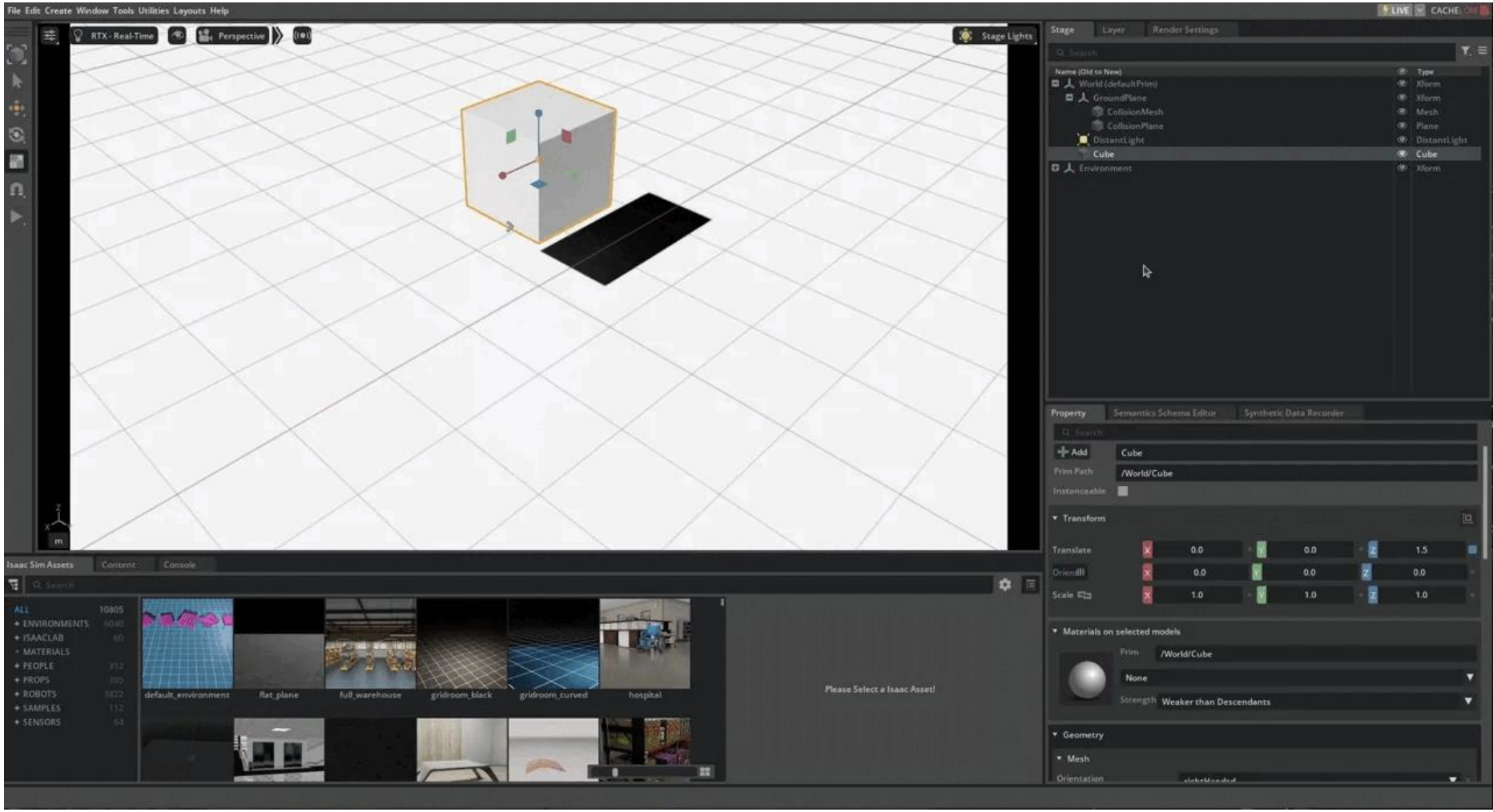
- **操作方式**：
  - 拖曳 **箭頭** → 沿單一軸移動 / 縮放
  - 拖曳 **彩色方塊** → 沿兩軸移動 / 縮放
  - 拖曳 **中心圓點 / 圓環** → 同時沿三軸移動 / 縮放




- **精確調整**：在 **Property Panel（屬性面板）** 中輸入數值修改。點擊數值旁的 **藍色方塊** 可重設為預設值。

# 新增物理和碰撞屬性 - GUI

- 右側 **Stage Tree** 中找到物件「**/World/Cube**」，並選取它
- 右下方的**屬性面板 (Property Panel)** 中，點擊 **Add** 按鈕，並從下拉選單中選擇 **Physics**
- 選擇 **Rigid Body with Colliders Preset (預設剛體與碰撞體)**
- 按下 **Play** 按鈕



# Extension

# Isaac Sim 基本使用 - Extension

這裡將示範 **Extension（擴充模組）**工作流程 的特性，使用一個現有的擴充模組 ——「**Script Editor**」

- **Script Editor** 允許使用者透過 **Python** 與 **Stage** 進行互動
- **開啟 Script Editor**：Window → Script Editor

**基本操作:**

1. **增加地平面:**

```
from isaacsim.core.api.objects.ground_plane import
GroundPlane GroundPlane(prim_path="/World/GroundPlane", z_position=0)
```

2. **增加光源:** 開啟 **Script Editor** 新分頁，在選單中點擊 **Tab → Add Tab**

```
import omni.usd
from pxr import Sdf, UsdLux
stage = omni.usd.get_context().get_stage()
distantLight = UsdLux.DistantLight.Define(stage, Sdf.Path("/DistantLight"))
distantLight.CreateIntensityAttr(300)
```

- **注意: 如果場景中存在光源，但沒有反射光線的物體，場景仍然會很暗。**

# Isaac Sim 基本使用 - Extension

3. 加入「可視化」立方體: 在**Script Editor** 開啟新分頁

- 直接生成立方體:

```
import numpy as np
from isaacsim.core.api.objects import VisualCuboid
VisualCuboid(
        prim_path="/visual_cube",
        name="visual_cube",
        position=np.array([0, 0.5, 0.5]),
        size=0.3,
        color=np.array([255, 255, 0]), )
VisualCuboid(
        prim_path="/test_cube",
        name="test_cube",
        position=np.array([0, -0.5, 0.5]),
        size=0.3,
        color=np.array([0, 255, 255]), )
```

新增兩個立方體: 純可視 & 預加入物理碰撞屬性

- 使用 **Isaac Sim Core** 原始 USD API 生成立方體 (更為冗長，但可以更好地控制每個屬性)

```
from pxr import UsdPhysics, PhysxSchema, Gf,
PhysicsSchemaTools, UsdGeom
import omni
# USD api for getting the stage
stage = omni.usd.get_context().get_stage()
# Adding a Cube
path = "/visual_cube_usd"
cubeGeom = UsdGeom.Cube.Define(stage, path)
cubePrim = stage.GetPrimAtPath(path)
size = 0.5
offset = Gf.Vec3f(1.5,-0.2,1.0)
cubeGeom.CreateSizeAttr(size)
if not cubePrim.HasAttribute("xformOp:translate"):
        UsdGeom.Xformable(cubePrim).AddTranslateOp().Set(offset)
else:
        cubePrim.GetAttribute("xformOp:translate").Set(offset)
```

# 移動、旋轉與縮放 - GUI

- 使用 core API 移動物件：

```
import numpy as np
from isaacsim.core.prims import XFormPrim
translate_offset = np.array([[1.5,1.2,1.0]])
orientation_offset = np.array([[0.7,0.7,0,1]])  # note this is in radians
scale = np.array([[1,1.5,0.2]])

stage = omni.usd.get_context().get_stage()
cube_in_coreapi = XFormPrim(prim_paths_expr="/test_cube")
cube_in_coreapi.set_world_poses(translate_offset, orientation_offset)
cube_in_coreapi.set_local_scales(scale)
```

# 移動、旋轉與縮放 - GUI

- 使用原始 USD API 移動物件：

```
from pxr import UsdGeom, Gf
import omni.usd
stage = omni.usd.get_context().get_stage()
cube_prim = stage.GetPrimAtPath("/visual_cube_usd")
translate_offset = Gf.Vec3f(1.5,-0.2,1.0)
rotate_offset = Gf.Vec3f(90,-90,180)   # note this is in degrees
scale = Gf.Vec3f(1,1.5,0.2)
# translation
if not cube_prim.HasAttribute("xformOp:translate"):
        UsdGeom.Xformable(cube_prim).AddTranslateOp().Set(translate_offset)
else:
        cube_prim.GetAttribute("xformOp:translate").Set(translate_offset)
# rotation
# there are also "xformOp:orient" for quaternion rotation, as well as "xformOp:rotateX", "xformOp:rotateY", "xformOp:rotateZ" for individual axis rotation
if not cube_prim.HasAttribute("xformOp:rotateXYZ"):
        UsdGeom.Xformable(cube_prim).AddRotateXYZOp().Set(rotate_offset)
else:
        cube_prim.GetAttribute("xformOp:rotateXYZ").Set(rotate_offset)
# scale
if not cube_prim.HasAttribute("xformOp:scale"):
        UsdGeom.Xformable(cube_prim).AddScaleOp().Set(scale)
else:
        cube_prim.GetAttribute("xformOp:scale").Set(scale)
```

# 新增物理和碰撞屬性 - Extension

**常見的物理屬性包括質量和慣性矩陣，它們使物體能夠在重力作用下下落。碰撞屬性使物體能夠與其他物體發生碰撞**

- 在 **Isaac Sim core API** 中，已經為常用的物件撰寫了**封裝類別 (wrapper)**，這些封裝會自動附帶所有必要的物理與碰撞屬性。

```
import numpy as np
from isaacsim.core.api.objects import DynamicCuboid
DynamicCuboid(
        prim_path="/dynamic_cube",
        name="dynamic_cube",
        position=np.array([0, -1.0, 1.0]),
        scale=np.array([0.6, 0.5, 0.2]),
        size=1.0,
        color=np.array([255, 0, 0]),
)
```

- 不使用封裝生成立方體，而是為前面生成的 **test_cube** 單獨賦予物理與碰撞屬性。

```
from isaacsim.core.prims import RigidPrim
RigidPrim("/test_cube")

from isaacsim.core.prims import GeometryPrim
prim = GeometryPrim("/test_cube")
prim.apply_collision_apis()
```

按下 Play 按鈕，即可看到立方體在重力作用下下落並與地平面碰撞。

# Standalone Python

# Isaac Sim 基本使用 - Standalone Python

- 在 **Isaac Sim 安裝包** 中尋找 **standalone_examples/tutorials/getting_started.py**
- 若要執行該腳本，請開啟終端，導覽至 Isaac Sim 安裝的根目錄，然後執行以下命令：

➢ Windows :

```
python.bat standalone_examples\tutorials\getting_started.py
```

➢ Linux:

```
./python.sh standalone_examples/tutorials/getting_started.py
```

**基本操作(getting_started.py):**

1. **增加地平面:**

```
from isaacsim.core.api.objects.ground_plane import GroundPlane
GroundPlane(prim_path="/World/GroundPlane", z_position=0)
```

2. **增加光源:**

```
from pxr import Sdf, UsdLux
stage = omni.usd.get_context().get_stage()
distantLight = UsdLux.DistantLight.Define(stage, Sdf.Path("/DistantLight"))
distantLight.CreateIntensityAttr(300)
```

# Isaac Sim 基本使用 - Standalone Python

3. **加入「可視化」立方體:** 在 getting_started.py 中將立方體加入場景中的程式碼行如下

```
import numpy as np
from isaacsim.core.api.objects import VisualCuboid
VisualCuboid(
prim_path="/visual_cube",
name="visual_cube",
position=np.array([0, 0.5, 0.5]),
size=0.3,
color=np.array([255, 255, 0]),
)
```

- 「可視化」立方體是指沒有附加物理屬性的立方體。沒有質量，也沒有碰撞。這種立方體不會因重力而下落，也不會與其他物體碰撞。可以按下「播放」按鈕，觀察模擬運行時立方體是否沒有任何反應。

# 移動、旋轉與縮放 - Standalone Python

- 以下的程式碼片段示範如何使用 **Isaac Sim core API (Core API)** 來移動場景中的物件：

```
import numpy as np
from isaacsim.core.prims.xform_prim import XformPrim
from isaacsim.core.prims.prim import Prim
translate_offset = np.array([[1.5,-0.2,1.0]])
rotate_offset = np.array([[90,-90,180]])
scale = np.array([[1,1.5,0.2]])

cube_in_coreapi = XformPrim(Prim(prim_paths_expr="/test_cube"))
cube_in_coreapi.set_world_poses(translate_offset, rotate_offset)
cube_in_coreapi.set_scales(scale)
```

# 新增物理和碰撞屬性 - Standalone Python

- 在 **Standalone Python 腳本** 中，可以透過將立方體轉換為 **RigidPrim 物件**（如下所示）來為其添加物理屬性。這賦予了物體在重力作用下下落的能力。

```
from isaacsim.core.prims import RigidPrim
RigidPrim("/visual_cube")
```

- 以下的程式碼片段展示了 **為物件新增碰撞屬性 (collision properties)** 的程式行，這些屬性讓物件能夠與其他物件發生碰撞互動。

```
from isaacsim.core.prims import GeometryPrim
prim = GeometryPrim("/visual_cube")
prim.apply_collision_apis()
```