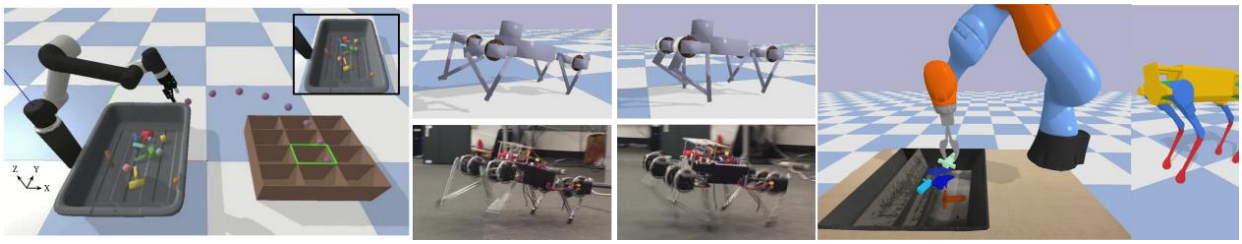
PyBullet 快速入门指南

Erwin Coumans, Yunfei Bai, 2016-2021
Visit desktop doc, forums and star Bullet!

目录

1. 简介

PyBullet 是一个快速且易于使用的用于机器人模拟和机器学习，专注于仿真到现实转换的 Python 模块. 使用 PyBullet，您可以从 URDF、SDF、MJCF 和其他文件格式加载关节体. PyBullet 提供正向动力学模拟、反向动力学计算、正向和反向运动学、碰撞检测和射线交叉查询功能。Bullet Physics SDK 中包括 PyBullet 机器人示例，例如模拟 Minitaur 四足机器人、使用 TensorFlow 推理运行的类人机器人和抓取物体的 KUKA 机械臂。 简化坐标多体、刚体和可变形体由统一的 LCP 约束求解器处理，类似于本论文.



除了物理模拟之外，还有渲染绑定，具有 CPU 渲染器 (TinyRenderer) 和 OpenGL 3.x 渲染和可视化，并支持虚拟现实耳机，如 HTC Vive 和 Oculus Rift。 PyBullet 还具有执行碰撞检测查询（最近点、重叠对、光线交叉测试等）和添加调试渲染（调试行和文本）的功能。PyBullet 具有对共享内存、UDP 和 TCP 网络的跨平台内置客户端-服务器支持。 因此，您可以在连接到 Windows VR 服务器的 Linux 上运行 PyBullet。

PyBullet 包装了新的Bullet C-API，它被设计为独立于底层物理引擎和渲染引擎，因此我们可以轻松迁移到 Bullet 的更新版本，或者使用不同的物理引擎或渲染引擎。默认情况下，PyBullet 在 CPU 上使用 Bullet 2.x API。我们还将使用 OpenCL 公开在 GPU 上运行的 Bullet 3.x。还有一个类似于 PyBullet 的 C++ API，参见b3RobotSimulatorClientAPI。

PyBullet 可以很容易地与 TensorFlow 和 OpenAI Gym 一起使用。 来自 Google Brain [1,2,3,4], X[1,2], Stanford AI Lab [1,2,3], OpenAI,INRIA [1] and many other labs 的研究人员都在使用 PyBullet. 如果您在研究中使用 PyBullet，请添加 引文.

PyBullet 的安装就像 (sudo) pip install PyBullet (Python 2.x)、pip3 install PyBullet 一样简单。这将公开 PyBullet 模块以及 pybullet_envs Gym 环境。

1.1 Hello PyBullet World

这是我们一步一步讨论的 PyBullet 介绍脚本：

```
   import   pybullet
as p import time
   import pybullet_data
   physicsClient  =  p.connect(p.GUI)#or  p.DIRECT  for  non-graphical
version p.setAdditionalSearchPath(pybullet_data.getDataPath()) #可选用
的ly p.setGravity(0,0,-10)
   planeId                        =
p.loadURDF("plane.urdf")  startPos
= [0,0,1]
   startOrientation  =   p.getQuaternionFromEuler([0,0,0])
boxId = p.loadURDF("r2d2.urdf",startPos, startOrientation)
   #set  the  center  of  mass  frame  (loadURDF  sets  base  link
frame)    startPos/Ornp.resetBasePositionAndOrientation(boxId,
startPos, startOrientation)
       for  i  in  range
    (10000):
    p.stepSimulation()
    time.sleep(1./240.)
   cubePos,                    cubeOrn                    =
p.getBasePositionAndOrientation(boxId)
print(cubePos,cubeOrn)
   p.disconnect()
```

1.2 connect, disconnect, bullet_client（连接、断开物理引擎，客户端 ）

导入 PyBullet 模块后，首先要做的是“连接”到物理模拟引擎。PyBullet 是围绕客户端-服务器驱动的 API 设计的，客户端发送命令，物理服务器返回状态。PyBullet 有一些内置的物理服务器：DIRECT 和 GUI。 GUI 和 DIRECT 连接都将在与 PyBullet 相同的过程中执行物理模拟和渲染。

请注意，在 DIRECT 模式下，您无法访问 OpenGL 和 VR 硬件功能，如“虚拟现实”和“调试 GUI、线条、文本、参数”章节中所述。 DIRECT 模式允许通过“getCameraImage”API 使用内置软件渲染器渲染图像。 这对于在没有 GPU 的服务器上在cloud中运行模拟非常有用。

您可以提供自己的数据文件，也可以使用 PyBullet 附带的 PyBullet_data 包。 为此，导入 pybullet_data 并使用 pybullet.setAdditionalSearchPath(pybullet_data.getDataPath()) 注册目录.

1.2.1 getConnectionInfo（获取连接信息）

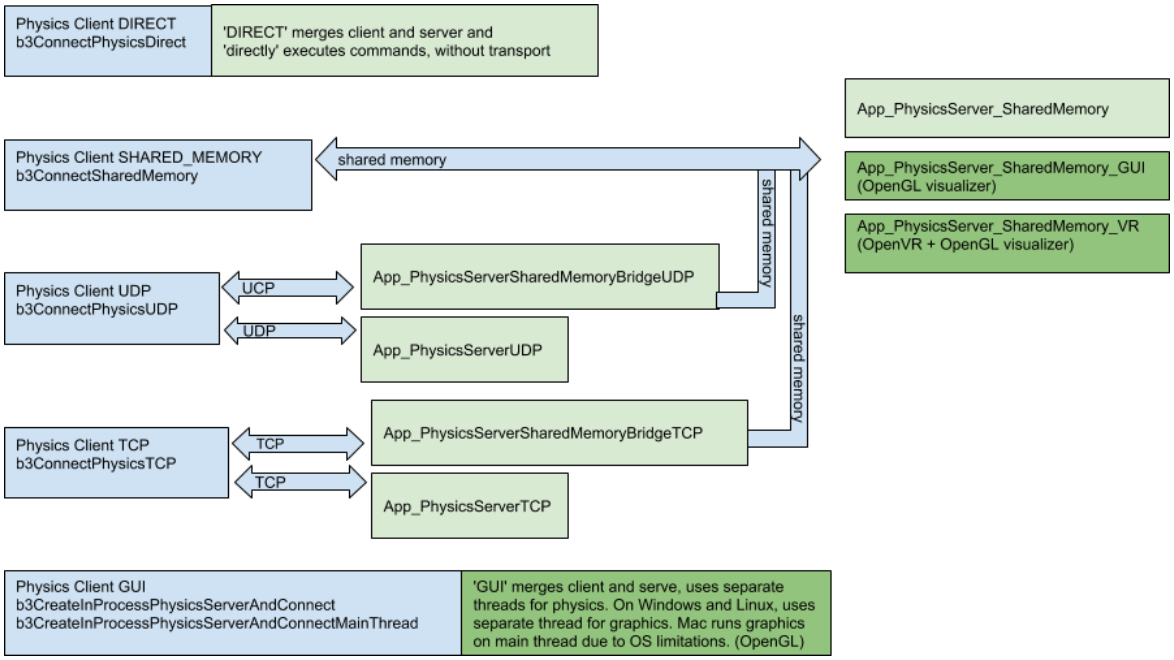
给定一个physicsClientId 将返回列表[isConnected, connectionMethod]

1.2.2 isConnected（已连接）

在给定一个 physicsClientId时如果已连接，isConnected 将返回 true，否则返回 false。

1.2.3 setTimeOut（设置超时）

如果服务器在特定超时值内未处理命令，则客户端将断开连接。 使用 setTimeOut 以秒为单位指定此值。



图中带有各种物理客户端（蓝色）和物理服务器（绿色）选项的图表。 深绿色服务器提供 OpenGL 调试可视化。

1.2.4 connect using DIRECT, GUI（使用 DIRECT、GUI 连接）

DIRECT连接直接向物理引擎发送命令，不使用任何传输层，也没有图形可视化窗口，执行命令后直接返回状态。

GUI 连接将在与 PyBullet 相同的进程空间内创建具有 3D OpenGL 渲染的新图形用户界面 (GUI)。 在 Linux 和 Windows 上，此 GUI 在单独的线程中运行，而在 OSX 上，由于操作系统限制，它在同一线程中运行。 在 Mac OSX 上，您可能会在 OpenGL 窗口中看到一个旋转轮，直到您运行 “stepSimulation”或其他 PyBullet 命令。

命令和状态消息使用普通内存缓冲区在 PyBullet 客户端和 GUI 物理模拟服务器之间发送。

还可以使用 SHARED_MEMORY、UDP 或 TCP 网络连接到同一台机器或远程机器上不同进程中的物理服务器。 有关详细信息，请参阅有关共享内存、UDP 和 TCP 的部分。

与几乎所有其他方法不同，由于向后兼容性，此方法不解析关键字参数。

连接输入参数是：

必需的	connection mode	integer: DIRECT, GUI, SHARED_MEMORY, UDP, TCP GUI_SERVER, SHARED_MEMORY_SERVER, SHARED_MEMORY_GUI	DIRECT 模式创建一个新的物理引擎并直接与其通信。GUI 将创建一个带有图形 GUI 前端的物理引擎并与之通信。 SHARED_MEMORY 将连接到同一台机器上现有的物理引擎进程，并通过共享内存与其通信。 TCP 或 UDP 将通过 TCP 或 UDP 网络连接到现有的物理服务器。 GUI_SERVER 类似于 GUI，但也充当允许外部 SHARED_MEMORY 连接的服务器。 SHARED_MEMORY_SERVER 类似于 DIRECT，但也充当允许外部 SHARED_MEMORY 连接的服务器。SHARED_MEMORY_GUI 类似于 DIRECT，但会尝试连接到外部图形服务器进行显示。 Bullet ExampleBrowser 可以选择充当物理服务器或图形服务器。
可选用的	key	int	在 SHARED_MEMORY 模式下，可选的共享内存密钥。启动 ExampleBrowser 或 SharedMemoryPhysics_* 时，您可以使用可选的命令行 --shared_memory_key 来设置密钥。 这允许在同一台机器上运行多个服务器。
可选用的	hostName (UDP and TCP)	string	IP 地址或主机名，例如“127.0.0.1”或“localhost”或“mymachine.domain.com”
可选用的	port (UDP and TCP)	integer	UDP 端口号。 默认 UDP 端口为 1234，默认 TCP 端口为 6667（与服务器中的默认值匹配）
可选用的	options	string	传递给 GUI 服务器的命令行选项。 您可以设置背景颜色，红/绿/蓝参数在 [0..1] 范围内，如下所示： p.connect(p.GUI,options="--background_color_red=1--background_color_blue=1 --background_color_green=1") 其他选项是: --mouse_move_multiplier=0.400000 (鼠标灵敏度) --mouse_wheel_multiplier=0.400000 (鼠标滚轮灵敏度) --width=<int>以像素为单位的窗口宽度 --height=<int>以像素为单位的窗口高度。 --mp4=moviename.mp4 (录像，需要 ffmpeg） --mp4fps=<int> (设置录像的每秒帧数).

如果未连接，connect 返回物理客户端 ID 或 -1。 物理客户端 Id 是大多数其他 PyBullet 命令的可选参数。 如果您不提供它，它将假定物理客户端 id = 0。您可以连接到多个不同的物理服务器，GUI 除外。

例如：

pybullet.connect(pybullet.DIRECT)

pybullet.connect(pybullet.GUI,options="--opengl2")

pybullet.connect(pybullet.SHARED_MEMORY,1234)

pybullet.connect(pybullet.UDP,"192.168.0.1")

pybullet.connect(pybullet.UDP,"localhost", 1234)

pybullet.connect(pybullet.TCP,"localhost", 6667)

1.2.5 connect using Shared Memory(利用共享内存连接)

有一些物理服务器允许共享内存连接： App_SharedMemoryPhysics 、App_SharedMemoryPhysics_GUI 和子弹示例浏览器在实验/物理服务器下有一个允许共享内存连接的示例。 这将使您可以在单独的过程中执行物理模拟和渲染。

您还可以通过共享内存连接到 App_SharedMemoryPhysics_VR，这是一款支持头戴式显示器和 6 自由度跟踪控制器的虚拟现实应用程序，例如 HTC Vive 和带有 Touch 控制器的 Oculus Rift。 由于 Valve OpenVR SDK 只能在 Windows 下正常工作，App_SharedMemoryPhysics_VR 只能在 Windows 下使用 premake（最好）或 cmake 构建。

1.2.6 connect using UDP or TCP networking（使用 UDP 或 TCP 网络连接）

对于 UDP 网络，有一个 App_PhysicsServerUDP 可以侦听某个 UDP 端口。 它使用开源的 enet 库来实现可靠的 UDP 网络。 这允许您在单独的机器上执行物理模拟和渲染。 对于 TCP，PyBullet 使用 clsocket 库。 这在使用从防火墙后面的机器到机器人模拟的 SSH 隧道时非常有用。例如，您可以在 Linux 上使用 PyBullet 运行控制堆栈或机器学习，同时使用 HTC Vive 或 Rift 在虚拟现实中的 Windows 上运行物理服务器。

另一个 UDP 应用程序是 App_PhysicsServerSharedMemoryBridgeUDP 应用程序，它充当现有物理服务器的桥接器：您可以通过 UDP 连接到此桥接器，桥接器使用共享内存连接到物理服务器：桥接器在客户端和服务器之间传递消息。 以类似的方式有一个 TCP 版本（用 TCP 替换 UDP）。

还有一个 GRPC 客户端和服务器支持，默认情况下不启用。 您可以使用 premake4 构建系统使用 --enable_grpc 选项进行尝试（请参阅 Bullet/build3/premake4）。

注意：目前，客户端和服务器都需要是 32 位或 64 位版本！

1.2.7 bullet_client（bullet客户端）

如果要并行使用多个独立的模拟，可以使用 pybullet_utils.bullet_client 。bullet_client.BulletClient(connection_mode=pybullet.GUI, options='') 的实例与 pybullet 实例具有相同的 API。 它会自动将适当的physicsClientId 添加到每个API 调用中。 PyBullet Gym 环境使用 bullet_client 来允许并行训练多个环境，请参阅 env_bases.py中的实现。另一个小例子展示了如何拥有两个单独的实例，每个实例都有自己的对象，请参见multipleScenes.py.

1.2.8 disconnect（断开链接）

您可以使用连接调用返回的物理客户端 ID（如果非负）与物理服务器断开连接。“DIRECT”或“GUI”物理服务器将关闭。 一个单独的（进程外）物理服务器将继续运行。 另请参阅“resetSimulation”以删除所有项目。

断开参数：

可选用的	physicsClientId	int	如果您连接到多个物理服务器，您可以选择哪一个。

1.3 setGravity(设置重力)

默认情况下，没有启用重力。 setGravity 允许您为所有对象设置默认重力。

setGravity （无返回值）输入参数为：

必要的	graX	float	沿 X 世界轴的重力
必要的	gravY	float	沿 Y 世界轴的重力
必要的	gravZ	float	沿Z 世界轴的重力
可选用的	physicsClientId	int	如果您连接到多个物理服务器，您可以选择哪一个。

1.4 loadURDF, loadSDF, loadMJCF(加载模型URDF、SDF、MJCF)

loadURDF 将向物理服务器发送命令以从 Universal Robot Description File (URDF) 加载物理模型。 URDF 文件被 ROS 项目（机器人操作系统）用来描述机器人和其他对象，它是由 WillowGarage 和开源机器人基金会（OSRF）创建的。 许多机器人都有公开的 URDF 文件，您可以在这里找到说明和教程： http://wiki.ros.org/urdf/Tutorials

重要说明：大多数关节（滑块、旋转、连续）默认启用电机以防止自由运动。 这类似于具有高摩擦谐波驱动的机器人关节。 您应该使用 pybullet.setJointMotorControl2 设置关节电机控制模式和目标设置。 有关更多信息，请参阅 setJointMotorControl2 API。

警告：默认情况下，PyBullet 会缓存一些文件以加快加载速度。 您可以使用 setPhysicsEngineParameter(enableFileCaching=0) 禁用文件缓存。

loadURDF 参数是：

必要的	fileName	string	物理服务器文件系统上 URDF 文件的相对或绝对路径。
可选用的	basePosition	vec3	在世界空间坐标 [X,Y,Z] 中的指定位置创建对象的基础。 请注意，此位置是 URDF 连杆位置。如果惯性系不为零，则这与质心位置不同。可以使用函数 resetBasePositionAndOrientation 设置质心位置/方向。
可选用的	baseOrientation	vec4	在指定方向创建对象的基础作为世界空间四元数 [X,Y,Z,W]。 请参阅 basePosition 中的注释。

可选用的	useMaximalCoordinates	int	实验性的。 默认情况下，URDF 文件中的关节是使用缩减坐标方法创建的：关节使用 Featherstone 铰接体算法（ABA，Bullet 2.x 中的 btMultiBody）进行模拟。 useMaximalCoordinates 选项将为每个连杆创建一个 6 自由度的刚体，这些刚体之间的约束用于对关节进行建模。
可选用的	useFixedBase	int	强制加载对象的基础是静态的
可选用的	flags	int	以下flag可以使用OR、| 组合起来： URDF_MERGE_FIXED_LINKS：这将从 URDF 文件中删除固定连杆并合并生成的连杆。 这对性能有好处，因为各种算法（关节体算法、正向运动学等）在关节数量上具有线性复杂性，包括固定关节。 URDF_USE_INERTIA_FROM_FILE：默认情况下，Bullet 根据碰撞形状的质量和体积重新计算惯性张量。 如果您可以提供更准确的惯性张量，请使用此标志。 URDF_USE_SELF_COLLISION：默认情况下，Bullet 禁用自碰撞。 这个标志让你启用它。 您可以使用以下标志自定义自碰撞行为： URDF_USE_SELF_COLLISION_INCLUDE_PARENT 将启用子件和父件之间的碰撞，默认情况下禁用。 需要与 URDF_USE_SELF_COLLISION 标志一起使用。 URDF_USE_SELF_COLLISION_EXCLUDE_ALL_PARENTS 将弃用子件与其任何祖先（父母，父母的父母，直到基础）之间的自碰撞。 需要配合使用

			URDF_USE_SELF_COLLISION. URDF_USE IMPLICIT CYLINDER，将使用平滑的隐式圆柱体。 默认情况下，Bullet 会将圆柱体细分为有限个凸包。 URDF_ENABLE_SLEEPING，允许在身体一段时间没有移动后禁用模拟。 与活动物体的交互将重新启用模拟。 URDF_INITIALIZE_SAT_FEATURES，将为凸面形状创建三角形网格。 这将改进可视化并且还允许使用分离轴测试 (SAT) 而不是 GJK/EPA。 需要使用 setPhysicsEngineParameter 启用 SAT。 URDF_USE_MATERIAL_COLORS_FROM_MTL，将使用来自 Wavefront OBJ 文件的 RGB 颜色，而不是来自 URDF 文件的颜色。 URDF_ENABLE_CACHED_GRAPHICS_SHAPES，将缓存和重用图形形状。 它将提高具有相似图形资产的文件的加载性能。 URDF_MAINTAIN_LINK_ORDER，将尝试维护来自 URDF 文件的连杆顺序。 说在URDF文件中，顺序是：ParentLink0、ChildLink1（附加到ParentLink0）、ChildLink2（附加到ParentLink0）。 如果没有这个标志，顺序可能是 P0、C2、C1。
可选用的	globalScaling	float	globalScaling 将对 URDF 模型应用比例因子。
可选用的	physicsClientId	int	如果您连接到多个服务器，您可以选择一个。

loadURDF 返回一个主体唯一 id，一个非负整数值。 如果无法加载 URDF 文件，则此整数将为负数且不是有效的body特有的 ID。

默认情况下，loadURDF 将使用凸包进行网格碰撞检测。对于静态(质量 = 0，不移动)网格，您可以通过在 URDF 中添加标签使网格凹入：

`<link concave="yes" name="baseLink">` 参见 samurai.urdf 的示例。 URDF 格式还有一些其他扩展，您可以浏览示例进行探索。 PyBullet 不会处理来自 URDF 文件的所有信息。 查看示例和 URDF 文件以了解支持哪些功能。 通常有一个 Python API 来控制该功能。 每个连杆只能有一种材料，因此如果您有多个不同材料的视觉形状，则需要将它们拆分为单独的连杆，通过固定关节连接。 您可以使用 OBJ2SDF 实用程序来执行此操作，这是 Bullet 的一部分。

1.4.2 loadSDF, loadMJCF（加载SDF，加载MJCF模型）

您还可以从其他文件格式加载对象，例如 .bullet、.sdf 和 .mjcf。 这些文件格式支持多个对象，因此返回值是一个对象唯一 ID 列表。 SDF 格式在 http://sdformat.org 中有详细说明。 loadSDF 命令只提取了 SDF 中与机器人模型和几何相关的一些基本部分，而忽略了许多与相机、灯光等相关的元素。 loadMJCF 命令执行 OpenAI Gym 中使用的 MuJoCo MJCF xml 文件的基本导入。 另请参阅 loadURDF 下与默认关节电机设置相关的重要说明，并确保使用 setJointMotorControl2。

必要的	fileName	string	物理服务器文件系统上 URDF 文件的相对或绝对路径。
可选用的	useMaximalCoordinates	int	实验性的。 有关更多详细信息，请参阅 loadURDF。
可选用的	globalScaling	float	SDF 和 URDF 支持 globalScaling，MJCF 不支持。 每个对象都将使用此比例因子（包括连杆、连杆框架、关节附件和线性关节限制）进行缩放。 这对质量没有影响，只对几何形状有影响。Use changeDynamics to change the mass if needed.
可选用的	physicsClientId	int	如果您连接到多个服务器，您可以选择一个。

loadBullet、loadSDF 和 loadMJCF 将返回一组对象唯一 ID：

objectUniqueIds（对象唯一 ID）	list of int	该列表包括加载的每个对象的对象唯一 ID。

1.5 saveState, saveBullet, restoreState（保存状态，保存bullet，恢复状态）

当您在恢复到先前保存的状态后需要确定性模拟时，需要存储所有重要的状态信息，包括接触点。 saveWorld 命令对此是不够的。 您可以使用 restoreState 命令从使用 saveState（内存中）或 saveBullet（磁盘上）拍摄的快照进行恢复。

saveState 命令仅将可选的 clientServerId 作为输入并返回状态 ID。 saveBullet 命令会将状态保存到磁盘上的 .bullet 文件中。

restoreState 命令输入参数为：

可选用的	fileName	string	使用 saveBullet 命令创建的 .bullet 文件的文件名。
可选用的	stateId	int	saveState 返回的状态 ID
可选用的	clientServerId	int	如果您连接到多台服务器，则可以选择一台

文件名或状态 ID 必须有效。 请注意，restoreState 会将对象的位置和关节角度重置为保存的状态，以及恢复接触点信息。 您需要确保在调用 restoreState 之前设置了对象和约束。 请参阅saveRestoreState.py 中的示例.

1.5.1 removeState（移除状态）

removeState 允许从内存中删除以前存储的状态。

1.5.2 saveWorld（储存World）

您可以将当前世界的近似快照创建为存储在服务器上的 PyBullet Python 文件。 saveWorld 可用作基本编辑功能，例如在 VR 中设置机器人、关节角度、对象位置和环境。 稍后您可以加载 PyBullet Python 文件来重新创建世界。 python 快照包含 loadURDF 命令以及关节角度和对象变换的初始化。 请注意，并非所有设置都存储在世界文件中。

输入参数是：

必要的	fileName	string	PyBullet 文件的文件名。
可选用的	clientServerId	int	如果您连接到多台服务器，则可以选择一台

1.6 createCollisionShape/VisualShape（创建碰撞形状/视觉形状）

虽然在世界上创建东西的推荐和最简单的方法是使用加载函数（loadURDF/SDF/MJCF/Bullet），但您也可以通过编程方式创建碰撞和视觉形状，并使用它们使用 createMultiBody 创建多体。 请参阅 Bullet Physics SDK 中的 createMultiBodyLinks.py 和 createVisualShape.py 示例。

createCollisionShape 的输入参数是

必要的	shapeType	int	GEOM_SPHERE, GEOM_BOX, GEOM_CAPSULE, GEOM_CYLINDER, GEOM_PLANE, GEOM_MESH, GEOM_HEIGHTFIELD
可选用的	radius	float	default 0.5: GEOM_SPHERE, GEOM_CAPSULE, GEOM_CYLINDER
可选用的	halfExtents	vec3 list of 3 floats	default [1,1,1]: for GEOM_BOX

可选用的	height	float	default: 1: for GEOM_CAPSULE, GEOM_CYLINDER
可选用的	fileName	string	Filename for GEOM_MESH, currently only Wavefront .obj. Will create convex hulls for each object (marked as 'o') in the .obj file.
可选用的	meshScale	vec3 list of 3 floats	default: [1,1,1], for GEOM_MESH
可选用的	planeNormal	vec3 list of 3 floats	default: [0,0,1] for GEOM_PLANE
可选用的	flags	int	GEOM_FORCE_CONCAVE_TRIMESH ：对于 GEOM_MESH，这将创建一个凹形静态三角形网格。 这不应与动态/移动对象一起使用，仅适用于静态（质量 = 0）地形。
可选用的	collisionFramePosition	vec3	碰撞形状相对于连杆框架的平移偏移
可选用的	collisionFrameOrientation	vec4	碰撞形状相对于连杆框架的旋转偏移（四元数 x,y,z,w）
可选用的	vertices	list of vec3	高度场的定义。 见 heightfield.py
可选用的	indices	list of int	高度场的定义
可选用的	heightfieldTextureScaling	float	高度场的纹理缩放
可选用的	numHeightfieldRows	int	高度场的定义
可选用的	numHeightfieldColumns	int	高度场的定义
可选用的	replaceHeightfieldIndex	int	替换现有的高度场（更新其高度）（比删除和重新创建高度场快得多）
可选用的	physicsClientId	int	如果您连接到多台服务器，则可以选择一台。

返回值是碰撞形状的非负 int 唯一 id，如果调用失败，则返回 -1。

1.6.1 createCollisionShapeArray（创建碰撞形状数组）

collisionShapeArray 是创建碰撞形状的数组版本。 有关用法，请参阅有关如何使用它的 snake.py 或 createVisualShapeArray.py 示例。

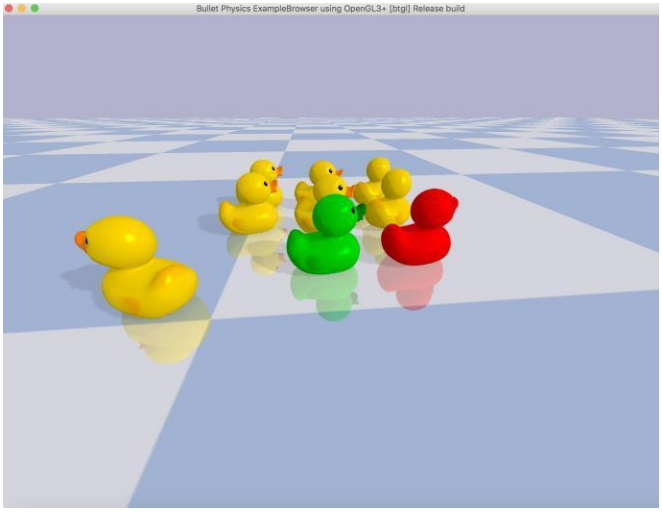
1.6.2 removeCollisionShape（移除碰撞形状）

removeCollisionShape 将删除现有的碰撞形状，使用其碰撞形状的唯一 ID.

1.6.3 createVisualShape（创建视觉形状）

您可以通过与创建碰撞形状类似的方式创建视觉形状，并使用一些额外的参数来控制视觉外观，例如漫反射和镜面反射颜色。 使用 GEOM_MESH 类型时，可以指向 Wavefront OBJ 文件，视觉形状会从材质文件 (.mtl) 中解析一些参数并加载纹理。 请注意，大纹理（超过 1024x1024 像素）会降低加载和运行时性能。

See examples/pybullet/examples/addPlanarReflection.py and createVisualShape.py



输入参数是

必要的	shapeType	int	GEOM_SPHERE, GEOM_BOX, GEOM_CAPSULE, GEOM_CYLINDER, GEOM_PLANE, GEOM_MESH
可选用的	radius	float	default 0.5: only for GEOM_SPHERE, GEOM_CAPSULE, GEOM_CYLINDER
可选用的	halfExtents	vec3 list of 3 floats	default [1,1,1]: only for GEOM_BOX
可选用的	length	float	default: 1: only for GEOM_CAPSULE, GEOM_CYLINDER (length = height)
可选用的	fileName	string	GEOM_MESH 的文件名，目前只有Wavefront .obj。将为 .obj 文件中的每个对象（标记为“o”）创建凸包。
可选用的	meshScale	vec3 list of 3 floats	default: [1,1,1],only for GEOM_MESH
可选用的	planeNormal	vec3 list of 3 floats	default: [0,0,1] only for GEOM_PLANE
可选用的	flags	int	unused / to be decided
可选用的	rgbaColor	vec4, list of 4 floats	红色、绿色、蓝色和 alpha 的颜色分量，每个都在 [0..1] 范围内。
可选用的	specularColor	vec3, list of 3 floats	镜面反射颜色，范围 [0..1] 内的红色、绿色、蓝色分量
可选用的	visualFramePosition	vec3, list of 3 floats	视觉形状相对于连杆框架的平移偏移

可选用的	vertices	list of vec3	您可以提供顶点、索引、坐标系和法线，而不是从 obj 文件创建网格。
可选用的	indices	list of int	三角形索引，应该是 3 的倍数
可选用的	uvs	list of vec2	顶点的 uv 纹理坐标。 使用 changeVisualShape 选择纹理图像。 uv的数量应该等于顶点的数量
可选用的	normals	list of vec3	顶点法线，数量应等于顶点数量.
可选用的	visualFrameOrientation	vec4, list of 4 floats	视觉形状相对于连杆框架的旋转偏移（四元数 x,y,z,w）
可选用的	physicsClientId	int	如果您连接到多台服务器，则可以选择一台。

返回值是视觉形状的非负 int 唯一 ID，如果调用失败，则返回 -1。 请参阅 createVisualShape, createVisualShapeArray 和 createTexturedMeshVisualShape 。

1.6.4 createVisualShapeArray（创建视觉形状数组）

createVisualShapeArray 是 createVisualShape 的数组版本。 请参阅 createVisualShapeArray.py 示例。

1.7 createMultiBody（创建多体）

尽管世界上创建东西最简单的方法是使用加载函数（loadURDF/SDF/MJCF/Bullet），但您可以使用 createMultiBody 创建多体。

请参阅 Bullet Physics SDK 中的 createMultiBodyLinks.py 示例。 createMultiBody 的参数与 URDF 和 SDF 参数非常相似。

您可以创建一个只有一个基础而没有关节/子连杆的多实体，也可以创建一个带有关节/子连杆的多实体。 如果您提供连杆，请确保每个列表的大小相同（len(linkMasses) == len(linkCollisionShapeIndices) 等）。 createMultiBody 的输入参数为：

可选用的	baseMass	float	底座质量，以 kg 为单位（如果使用 SI 单位）
可选用的	baseCollisionShapeIndex	int	来自 createCollisionShape 的唯一 ID 或 -1。 您可以将碰撞形状重新用于多个多体（实例化）
可选用的	baseVisualShapeIndex	int	来自 createVisualShape 的唯一 ID 或 -1。 您可以重用视觉形状（实例化）

可选用的	basePosition	vec3, list of 3 floats	底的笛卡尔世界位置
可选用的	baseOrientation	vec4, list of 4 floats	碱基的方向为四元数 [x,y,z,w]
可选用的	baseInertialFramePosition	vec3, list of 3 floats	惯性系局部位置
可选用的	baseInertialFrameOrientation	vec4, list of 4 floats	惯性系的局部方向，[x,y,z,w]
可选用的	linkMasses	list of float	质量值列表，每个连杆一个。
可选用的	linkCollisionShapeIndices	list of int	唯一 id 列表，每个连杆一个。
可选用的	linkVisualShapeIndices	list of int	每个连杆的视觉形状唯一 ID 列表
可选用的	linkPositions	list of vec3	相对于父级的本地连杆位置列表
可选用的	linkOrientations	list of vec4	本地连杆方向列表，w.r.t. 父件
可选用的	linkInertialFramePositions	list of vec3	局部惯性坐标系 pos 列表。 在连杆框架中
可选用的	linkInertialFrameOrientations	list of vec4	局部惯性系列表。 在连杆框架中
可选用的	linkParentIndices	list of int	父连杆的链接索引或基数为 0。
可选用的	linkJointTypes	list of int	关节类型列表，每个连杆一个。 目前支持 JOINT_REVOLUTE 、 JOINT_PRISMATIC 、 JOINT_SPHERICAL 和 JOINT_FIXED 类型。
可选用的	linkJointAxis	list of vec3	局部坐标系中的关节轴
可选用的	useMaximalCoordinates	int	实验性的，最好将其保留为 0/false。
可选用的	flags	int	类似于在 loadURDF 中传递的标志，例如 URDF_USE_SELF_COLLISION。 有关标志说明，请参见 loadURDF。
可选用的	batchPositions	list of vec3	基本位置数组，用于快速批量创建许多多体。参见example.。
可选用的	physicsClientId	int	如果您连接到多台服务器，您可以选择一台.

createMultiBody 的返回值是一个非负的唯一 id 或 -1 表示失败。 例子：

cuid = pybullet.createCollisionShape(pybullet.GEOM_BOX, halfExtents = [1, 1, 1]) mass= 0 #static box

pybullet.createMultiBody(mass,cuid)

See also createMultiBodyLinks.py, createObstacleCourse.py and createVisualShape.py in the Bullet/examples/pybullet/examples folder.

1.7.1 getMeshData（获取网格数据）

getMeshData 是一个实验性的未记录 API，用于返回三角形网格的网格信息（顶点、索引）。

必要的	bodyUniqueId	int	body 特有的 id
可选用的	linkIndex	int	链接索引
可选用的	collisionShapeIndex	int	复合形状的索引，如果连杆中有多个碰撞形状（请参阅 getCollisionShapeData）
可选用的	flags	int	默认情况下，PyBullet 将返回图形渲染顶点。 由于复制了具有不同法线的顶点，因此可以比原始网格中的顶点更多。 您可以使用 flags = pybullet.MESH_DATA_SIMULATION_MESH 接收模拟顶点
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

1.8 stepSimulation（逐步模拟）

stepSimulation 将在单个正向动力学仿真步骤中执行所有操作，例如碰撞检测、约束求解和集成。 默认时间步长为 1/240 秒，可以使用 setTimeStep 或 setPhysicsEngineParameter API 更改。

stepSimulation 输入参数是可选的：

可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

默认情况下，stepSimulation 没有返回值。

仅用于实验/高级用途：如果通过 setPhysicsEngineParameter API 启用 reportSolverAnalytics，则以下信息将作为岛信息列表返回，其中包含以下详细信息：

islandId	int	island unique id
numBodies	list of body unique ids	the body unique ids in this island
numIterationsUsed	int	the number of solver iterations used.
remainingResidual	float	the residual constraint error.

另请参阅 setRealTimeSimulation 以自动让物理服务器根据其实时时钟运行正向动力学模拟。

1.8.1 setRealTimeSimulation（设置实时模拟）

默认情况下，物理服务器不会步进模拟，除非您明确发送“stepSimulation”命令。 这样，您可以保持模拟的控制确定性。 可以通过使用 setRealTimeSimulation 命令让物理服务器根据其实时时钟 (RTC) 自动步进仿真来实时运行仿真。 如果启用实时模拟，则无需调用“stepSimulation”。

请注意，setRealTimeSimulation 在 DIRECT 模式下无效：在 DIRECT 模式下，物理服务器和客户端发生在同一个线程中，并且您触发每个命令。 在 GUI 模式、虚拟现实模式和 TCP/UDP 模式下，物理服务器在与客户端 (PyBullet) 不同的线程中运行，setRealTimeSimulation 允许物理服务器线程添加对 stepSimulation 的额外调用。

输入参数为：

必要的	enableRealTimeSimulation	int	0 禁用实时仿真，1 启用
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

1.9 getBasePositionAndOrientation（获取基本位置和方向）

getBasePositionAndOrientation 在笛卡尔世界坐标中报告身体的基部（或根连杆）的当前位置和方向。 方向是 [x,y,z,w] 格式的四元数。

getBasePositionAndOrientation 输入参数为：

必要的	objectUniqueId	int	对象唯一 ID，从 loadURDF 返回。
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

getBasePositionAndOrientation 以 [x,y,z,w] 顺序返回 3 个浮点数的位置列表和方向作为 4 个浮点数的列表。 如果需要，使用 getEulerFromQuaternion 将四元数转换为 Euler。

另请参见 resetBasePositionAndOrientation 以重置对象的位置和方向。

这样就完成了第一个 PyBullet 脚本。 Bullet 在 Bullet/data 文件夹中附带了几个 URDF 文件。

1.10 resetBasePositionAndOrientation（重置基本位置和方向）

您可以重置每个对象的基础（根）的位置和方向。 最好只在开始时执行此操作，而不是在运行模拟期间执行此操作，因为该命令将覆盖所有物理模拟的效果。 线速度和角速度设置为零。您可以使用 resetBaseVelocity 重置为非零线性和/或角速度。

resetBasePositionAndOrientation 的输入参数是：

必要的	bodyUniqueId	int	对象唯一 ID，从 loadURDF 返回。
必要的	posObj	vec3	在世界空间坐标 [X,Y,Z] 中的指定位置重置对象的基础
必要的	ornObj	vec4	将对象在指定方向的底部重置为世界空间四元数 [X,Y,Z,W]
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

没有返回参数。

1.11 Transforms: Position and Orientation（变换：位置和方向）

物体的位置可以用笛卡尔世界空间坐标[x,y,z]表示。 可以使用四元数 [x,y,z,w]、欧拉角 [yaw、pitch、roll] 或 3x3 矩阵来表示对象的方向（或旋转）。 PyBullet 提供了一些辅助函数来在四元数、欧拉角和 3x3 矩阵之间进行转换。 此外，还有一些乘法和反转变换的函数。

1.11.1 getQuaternionFromEuler and getEulerFromQuaternion（从欧拉角得到四元数组并从四元数组得到欧拉角）

PyBullet API 使用四元数来表示方向。 由于四元数对人们来说不是很直观，因此有两个 API 可以在四元数和欧拉角之间进行转换。

getQuaternionFromEuler 输入参数是：

必要的	eulerAngle	vec3: list of 3 floats	X、Y、Z 欧拉角以弧度为单位，累积 3 次旋转，表示围绕 X 轴的滚动、围绕 Y 轴的俯仰和围绕 Z 轴的偏航。
可选用的	physicsClientId	Int	未使用，为 API 一致性而添加。

getQuaternionFromEuler 返回 4 个浮点值 [X,Y,Z,W] 的四元数 vec4 列表。

1.11.2 getEulerFromQuaternion（从四元数组中得到欧拉角）

getEulerFromQuaternion 输入参数是：

必要的	quaternion	vec4: list of 4 floats	四元数格式为 [x,y,z,w]
可选用的	physicsClientId	int	未使用，为 API 一致性而添加。

getEulerFromQuaternion 返回 3 个浮点值的列表，即 vec3。 旋转顺序是首先绕 X 滚动，然后绕 Y 俯仰，最后绕 Z 偏航，就像在 ROS URDF rpy 约定中一样。

1.11.3 getMatrixFromQuaternion（从四元数组获取矩阵）

getMatrixFromQuaternion 是一个实用 API，用于从四元数创建 3x3 矩阵。 输入是一个四元数，输出一个包含 9 个浮点数的列表，表示矩阵。

1.11.4 getAxisAngleFromQuaternion（从四元数组获取轴向角）

getAxisAngleFromQuaternion 将返回给定四元数方向的轴和角度表示。

必要的	quaternion	list of 4 floats	方向
可选用的	physicsClientId	int	未使用，为 API 一致性而添加。

1.11.5 multiplyTransforms, invertTransform（乘法变换，反转变换）

PyBullet 提供了一些辅助函数来进行乘法和逆变换。 这有助于将坐标从一个坐标系转换到另一个坐标系。

multiplyTransforms 的输入参数为：

必要的	positionA	vec3, list of 3 floats	
必要的	orientationA	vec4, list of 4 floats	四元数组 [x,y,z,w]
必要的	positionB	vec3, list of 3 floats	
必要的	orientationB	vec4, list of 4 floats	四元数组[x,y,z,w]
可选用的	physicsClientId	int	未使用，为 API 一致性而添加。

返回值是位置 (vec3) 和方向 (vec4, 四元数 x,y,x,w) 的列表。 invertTransform 的输入和输出参数为：

必要的	position	vec3, list of 3 floats	
必要的	orientation	vec4, list of 4 floats	四元数组[x,y,z,w]

invertTransform 的输出是位置 (vec3) 和方向 (vec4, 四元数 x,y,x,w)。

1.11.6 getDifferenceQuaternion（得到差分四元数）

getDifferenceQuaternion 将返回一个从开始方向到结束方向进行插值的四元数。

必要的	quaternionStart	list of 4 floats	开始方向
必要的	quaternionEnd	list of 4 floats	末端方向
必要的	physicsClientId	int	未使用，为 API 一致性而添加。

1.11.7 getAPIVersion（获取 API 版本）

您可以按年-月-0-天格式查询API版本。 您只能在相同 API 版本、相同位数（32 位/64 位）的物理客户端/服务器之间连接.

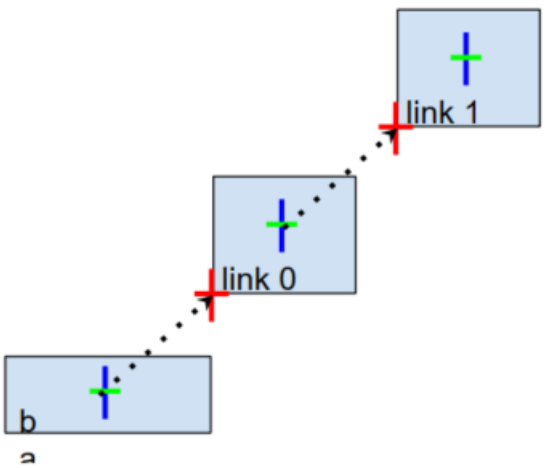
为了 API 的一致性，添加了一个可选的未使用参数physicsClientId。

可选用的	physicsClientId	Int	未使用，为 API 一致性而添加。

2. Controlling a robot（控制机器人）

在简介中，我们已经展示了如何初始化 PyBullet 和加载一些对象。 如果将 loadURDF 命令中的文件名替换为“r2d2.urdf”，则可以模拟 ROS 教程中的 R2D2 机器人。 让我们控制这个 R2D2 机器人移动、环顾和控制夹爪。 为此，我们需要知道如何访问其关节电机。

2.1 Base, Joints, Links（基座、关节、连杆）



URDF 文件中描述的模拟机器人有一个底座，以及通过关节连接的可选连杆。 每个关节将一个父连杆连接到一个子连杆。 在层次结构的根部，有一个我们称之为基础的根父级。 底座可以是完全固定的，自由度为 0，也可以是完全自由的，自由度为 6。 由于每个连杆都通过单个关节连接到父节点，因此关节的数量等于连杆的数量。 常规连杆的链接索引范围为 [0..getNumJoints()] 由于基础不是常规“连杆”，因此我们使用 -1 的约定作为其链接索引。 我们使用的惯例是关节框架相对于父惯性框架的质心中心表示，它与惯性主轴对齐。

2.2 getNumJoints, getJointInfo（获取关节数，获取关节信息）

加载机器人后，您可以使用 getNumJoints API 查询关节数。 对于 r2d2.urdf 这应该返回 15。

getNumJoints 输入参数：

必要的	bodyUniqueId	int	由 loadURDF 等返回的主体唯一 ID。
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

getNumJoints 返回一个整数值，表示关节的数量。

2.2.1 getJointInfo（获取关节信息）

对于每个关节，我们可以查询一些信息，例如它的名称和类型。 getJointInfo 输入参数

必要的	bodyUniqueId	int	由 loadURDF 等返回的主体唯一 ID。
必要的	jointIndex	int	[0 .. getNumJoints(bodyUniqueId)) 范围内的索引
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

getJointInfo 返回信息列表：

jointIndex	int	与输入参数相同的关节索引
jointName	string	关节的名称，在 URDF（或 SDF 等）文件中指定
jointType	int	关节的类型，这也意味着位置和速度变量的数量。 JOINT_REVOLUTE、JOINT_PRISMATIC、JOINT_SPHERICAL、JOINT_PLANAR、JOINT_FIXED。 有关更多详细信息，请参阅关于基础、接头和链接的部分。
qIndex	int	此主体的位置状态变量中的第一个位置索引
uIndex	int	该物体的速度状态变量中的第一个速度指数
flags	int	预订的
jointDamping	float	URDF 文件中指定的关节阻尼值
jointFriction	float	URDF 文件中指定的关节摩擦值
jointLowerLimit	float	滑块和旋转（铰链）关节的位置下限。
jointUpperLimit	float	滑块和旋转关节的位置上限。 在上限 < 下限的情况下忽略值。
jointMaxForce	float	URDF 中指定的最大力（可能是其他文件格式）请注意，此值不会自动使用。您可以在“setJointMotorControl2”中使用 maxForce。
jointMaxVelocity	float	URDF 中指定的最大速度。 请注意，目前实际电机控制命令中并未使用最大速度。
linkName	string	连杆的名称，在 URDF（或 SDF 等）文件中指定
jointAxis	vec3	局部框架中的关节轴（对于 JOINT_FIXED 忽略）
parentFramePos	vec3	父框架中的关节位置
parentFrameOrn	vec4	父框架中的关节方向（四元数 x,y,z,w）
parentIndex	int	父链接索引，-1 为基础

2.3 setJointMotorControl2/Array（设置关节电机控制 2/阵列）

注意： setJointMotorControl 已过时并由 setJointMotorControl2 API 取代。 （或者甚至更好地使用 setJointMotorControlArray）。

我们可以通过为一个或多个关节电机设置所需的控制模式来控制机器人。 在 stepSimulation 期间，物理引擎将模拟电机以达到给定的目标值，该目标值可以在最大电机力和其他约束条件下达到。

重要提示：默认情况下，每个旋转关节和棱柱关节都使用速度电机进行电动化。 您可以使用最大驱动力 0 禁用这些默认电机。这将让您执行扭矩控制。

For example:

```
maxForce = 0
mode = p.VELOCITY_CONTROL
    p.setJointMotorControl2(objUid, jointIndex,
  controlMode=mode, force=maxForce)
```

您还可以使用小的非零力来模拟关节摩擦。

如果您希望车轮保持恒定速度，您可以使用最大力：

```
maxForce = 500
p.setJointMotorControl2(bodyUniqueId=objUid
,jointIndex=0, controlMode=p.VELOCITY_CONTROL,
targetVelocity = targetVel, force = maxForce)
```

setJointMotorControl2 的输入参数是：

必要的	bodyUniqueId	int	从 loadURDF 等返回的正文唯一 ID。
必要的	jointIndex	int	范围 [0..getNumJoints(bodyUniqueId) 中的链接索引（请注意，链接索引 == 关节索引）
必要的	controlMode	int	POSITION_CONTROL（实际上是 CONTROL_MODE_POSITION_VELOCITY_PD）、VELOCITY_CONTROL、TORQUE_CONTROL 和 PD_CONTROL。（还有用于稳定（隐式）PD 控制的实验性 STABLE_PD_CONTROL，这需要额外的准备。有关 STABLE_PD_CONTROL 示例，请参见 humanoidMotionCapture.py 和 pybullet_envs.deep_mimc。）TORQUE_CONTROL 将立即应用扭矩，因此它仅在显式调用 stepSimulation 时有效。
可选用的	targetPosition	float	在 POSITION_CONTROL 中，targetValue 是关节的目标位置
可选用的	targetVelocity	float	在 VELOCITY_CONTROL 和 POSITION_CONTROL 中，targetVelocity 是关节的期望速度，请参见下面的实现说明。 请注意，targetVelocity 不是最大关节速度。 在 PD_CONTROL 和 POSITION_CONTROL/CONTROL_MODE_POSITION_VELOCITY_PD 中，使用以下公式计算最终目标速度：kp*(erp*(desiredPosition-currentPosition)/dt)+currentVelocity+kd*(m_desiredVelocity - currentVelocity) 。 另请参见示例 /pybullet/examples/pdControl.py

可选用的	force	float	在 POSITION_CONTROL 和 VELOCITY_CONTROL 中，这是用于达到目标值的最大电机力。 在 TORQUE_CONTROL 中，这是每个模拟步骤要应用的力/扭矩。
可选用的	positionGain	float	请参阅下面的实施说明
可选用的	velocityGain	float	请参阅下面的实施说明
可选用的	maxVelocity	float	在 POSITION_CONTROL 中，这将速度限制为最大值
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

注意：关节电机控制器的实际实现对于 POSITION_CONTROL 和 VELOCITY_CONTROL 是作为约束，对于 TORQUE_CONTROL 是作为外力：

method	implementation	component	约束误差最小化
POSITION_CONTROL	constraint	velocity and position constraint	error = position_gain*(desired_position-actual_position)+velocity_gain*(desired_velocity-actual_velocity)
VELOCITY_CONTROL	constraint	pure velocity constraint	error = desired_velocity - actual_velocity
TORQUE_CONTROL	external force		

通常最好从 VELOCITY_CONTROL 或 POSITION_CONTROL 开始。 执行 TORQUE_CONTROL（力控制）要困难得多，因为模拟正确的力依赖于非常准确的 URDF/SDF 文件参数和系统识别（正确的质量、惯性、质心位置、关节摩擦等）。

2.3.1 setJointMotorControlArray（设置联合电机控制阵列）

您可以为所有输入传递数组，而不是对每个关节进行单独调用，以显着减少调用开销。

setJointMotorControlArray 采用与 setJointMotorControl2 相同的参数，除了用整数列表替换整数。

setJointMotorControlArray 的输入参数是：

必要的	bodyUniqueId	int	从 loadURDF 等返回的正文唯一 ID.
必要的	jointIndices	list of int	[0..getNumJoints(bodyUniqueId) 范围内的索引（请注意，链接索引 == 关节索引）
必要的	controlMode	int	位置控制，速度控制，TORQUE_CONTROL，PD_CONTROL。 （还有用于稳定（隐式）PD 控制的实验性 STABLE_PD_CONTROL，这需要额外的准备。有关 STABLE_PD_CONTROL 示例，请参见 humanoidMotionCapture.py 和 pybullet_envs.deep_mimc。）

可选用的	targetPositions	list of float	在 POSITION_CONTROL 中，targetValue 是关节的目标位置
可选用的	targetVelocities	list of float	在 PD_CONTROL、VELOCITY_CONTROL 和 POSITION_CONTROL targetValue 是关节的目标速度，参见下面的实现说明。
可选用的	forces	list of float	在 PD_CONTROL、POSITION_CONTROL 和 VELOCITY_CONTROL 这是用于达到目标值的最大电机力。 在 TORQUE_CONTROL 中，这是每个模拟步骤要应用的力/扭矩。
可选用的	positionGains	list of float	请参阅下面的实施说明
可选用的	velocityGains	list of float	请参阅下面的实施说明
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

有关使用 setJointMotorControlArray 的示例，请参见

bullet3/examples/pybullet/tensorflow/humanoid_running.py。

2.3.2 setJointMotorControlMultiDof（设置联合电机控制多自由度）

setJointMotorControlMultiDof 与 setJointMotorControl2 类似，但它支持球面（multiDof）关节，这用于 deep_mimic 环境（在 pybullet_envs 中）和 humanoidMotionCapture.py 示例。targetPosition、targetVelocity 和 force 参数不是单个浮点数，而是接受 1 个浮点数的列表或 3 个浮点数的列表来支持球形关节。

setJointMotorControlMultiDof 的输入参数是：

必要的	bodyUniqueId	int	从 loadURDF 等返回的正文唯一 ID。
必要的	jointIndex	int	范围 [0..getNumJoints(bodyUniqueId) 中的链接索引（请注意，链接索引 == 关节索引）
必要的	controlMode	int	POSITION_CONTROL（实际上是 CONTROL_MODE_POSITION_VELOCITY_PD ）、VELOCITY_CONTROL、TORQUE_CONTROL 和 PD_CONTROL。（还有用于稳定（隐式）PD 控制的实验性 STABLE_PD_CONTROL，这需要额外的准备。有关 STABLE_PD_CONTROL 示例，请参见 humanoidMotionCapture.py 和 pybullet_envs.deep_mimc。）
可选用的	targetPosition	list of 1 or 3 floats	在 POSITION_CONTROL 中，targetValue 是关节的目标位置
可选用的	targetVelocity	list of 1 or 3 floats	在 VELOCITY_CONTROL 和 POSITION_CONTROL 中，targetVelocity 是关节的期望速度，请参见下面的实现说明。 请注意，targetVelocity 不是最大关节速度。 在 PD_CONTROL 和 POSITION_CONTROL/CONTROL_MODE_POSITION_VELOC

			ITY_PD 中，使用以下公式计算最终目标速度：kp*(erp*(desiredPosition-currentPosition)/dt)+currentVelocity+kd*(m_desiredVelocity - currentVelocity)。另请参见示例/pybullet/examples/pdControl.py the maximum joint velocity. In PD_CONTROL and
可选用的	force	list of 1 or 3 floats	在 POSITION_CONTROL 和 VELOCITY_CONTROL 中，这是用于达到目标值的最大电机力。在 TORQUE_CONTROL 中，这是每个模拟步骤要应用的力/扭矩。
可选用的	positionGain	float	请参阅下面的实施说明
可选用的	velocityGain	float	请参阅下面的实施说明
可选用的	maxVelocity	float	在 POSITION_CONTROL 中，这将速度限制为最大值
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

2.3.3 setJointMotorControlMultiDofArray（设置联合电机控制多自由度阵列）

setJointMotorControlMultiDofArray 是 setJointMotorControlMultiDof 的更高效版本，传入多个控制目标以避免/减少 Python 和 PyBullet C++ 扩展之间的调用开销。有关示例，请参见 humanoidMotionCapture.py。

setJointMotorControlMultiDofArray 的输入参数是：

必要的	bodyUniqueId	int	从 loadURDF 等返回的正文唯一 ID
必要的	jointIndices	list of int	范围 [0..getNumJoints(bodyUniqueId) 中的链接索引（请注意，连杆索引 == 关节索引）
必要的	controlMode	int	POSITION_CONTROL（实际上是 CONTROL_MODE_POSITION_VELOCITY_PD）、VELOCITY_CONTROL、TORQUE_CONTROL 和PD_CONTROL。（还有用于稳定（隐式）PD 控制的实验性 STABLE_PD_CONTROL，这需要额外的准备。有关 STABLE_PD_CONTROL 示例，请参见 humanoidMotionCapture.py 和 pybullet_envs.deep_mimc。）
可选用的	targetPositions	list of float or vec3	在 POSITION_CONTROL 中，targetValue 是关节的目标位置
可选用的	targetVelocities	list of float	在 VELOCITY_CONTROL 和 POSITION_CONTROL 中，targetVelocity 是关节的期望速度，请参见下面的实现说明。请注意，targetVelocity 不是最大关节速度。在 PD_CONTROL 和 POSITION_CONTROL/CONTROL_MODE_POSITION_VELOCITY_PD 中，使用以下公式计算最终目标速度：kp*(erp*(desiredPosition-currentPosition)/dt)+currentVelocity+kd*(m_desiredVelocity - currentVelocity)。另请参见示例 /pybullet/examples/pdControl.py
可选用的	forces	list of float	在 POSITION_CONTROL 和 VELOCITY_CONTROL 中，这是用于达到目标值的最大电机力。在 TORQUE_CONTROL 中，这是每个模拟步骤要应用的力/扭矩。
可选用的	positionGains	list of float	请参阅 setJointMotorControl2 中的实现说明
可选用的	velocityGains	list of float	请参阅 setJointMotorControl2 中的实现说明
可选用的	maxVelocities	list of float	在 POSITION_CONTROL 中，这将速度限制为最大值
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

2.4 getJointState(s), resetJointState（获取关节状态，重置关节状态）

我们可以使用 getJointState 从关节中查询多个状态变量，例如关节位置、速度、关节反作用力和关节电机扭矩。

getJointState 输入参数

必要的	bodyUniqueId	int	loadURDF 等返回的正文唯一 ID
必要的	jointIndex	int	范围 [0..getNumJoints(bodyUniqueId)] 中的链接索引
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

getJointState 输出

jointPosition	float	此关节的位置值。
jointVelocity	float	此关节的速度值。
jointReactionForces	list of 6 floats	这些是关节反作用力，如果为此关节启用了扭矩传感器，则为 [Fx, Fy, Fz, Mx, My, Mz]。 如果没有扭矩传感器，则为 [0,0,0,0,0,0]。
appliedJointMotorTorque	float	这是在最后一步模拟期间应用的电机扭矩。 请注意，这只适用于 VELOCITY_CONTROL 和 POSITION_CONTROL。 如果您使用 TORQUE_CONTROL，则应用的关节电机扭矩正是您提供的，因此无需单独报告。

2.4.1 getJointStates（获取关节状态）

getJointStates 是 getJointState 的数组版本。 不是传入单个jointIndex，而是传入一个 jointIndices 列表。

2.4.2 getJointStateMultiDof（获得关节状态多自由度）

还有用于球形关节的 getJointStateMultiDof。getJointState 输出

jointPosition	list of 1 or 4 float	此关节的位置值（作为关节角/位置或关节方向四元数）
jointVelocity	list of 1 or 3 float	此关节的速度值。
jointReactionForces	list of 6 floats	这些是关节反作用力，如果为此关节启用了扭矩传感器，则为 [Fx, Fy, Fz, Mx, My, Mz]。 如果没有扭矩传感器，则为 [0,0,0,0,0,0]。
appliedJointMotorTorque	float	这是在最后一步模拟期间应用的电机扭矩。 请注意，这只适用于 VELOCITY_CONTROL 和 POSITION_CONTROL。 如果您使用 TORQUE_CONTROL，则应用的关节电机扭矩正是您提供的，因此无需单独报告。

2.4.3 getJointStatesMultiDof（获得关节状态多自由度）

getJointStatesMultiDof 允许查询多个关节状态，包括多自由度（球形）关节。

2.4.4 resetJointState（重置关节状态）

您可以重置关节的状态。 最好只在开始时执行此操作，而不是运行模拟：resetJointState 会覆盖所有物理模拟。 请注意，我们目前仅支持 1-DOF 电动关节，滑动关节或旋转关节。

必要的	bodyUniqueId	int	loadURDF 等返回的正文唯一 ID
必要的	jointIndex	int	范围内的联合索引 [0..getNumJoints(bodyUniqueId)]
必要的	targetValue	float	关节位置（弧度或位置角度）
可选用的	targetVelocity	float	关节速度（角速度或线速度）
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

2.4.5 resetJointState(s)MultiDof（重置关节状态的多自由度）

还有用于球形关节的 resetJointStateMultiDof。 有关 resetJointStateMultiDof 的示例，请参见humanoidMotionCapture 。 还有一个 resetJointStatesMultiDof 一次重置多个关节。

2.5 enableJointForceTorqueSensor（启用关节力/扭矩传感器）

您可以启用或禁用每个关节中的关节力/扭矩传感器。 启用后，如果您执行 stepSimulation，“getJointState”将报告固定自由度中的关节反作用力：固定关节将测量所有 6DOF 关节力/扭矩。旋转/铰链关节力/扭矩传感器将沿除铰链轴之外的所有轴测量 5DOF 反作用力。 关节电机施加的力在 getJointState 的 AppliedJointMotorTorque 中可用。

enableJointForceTorqueSensor 的输入参数是：

必要的	bodyUniqueId	int	loadURDF 等返回的正文唯一 ID
必要的	jointIndex	int	范围内的联合索引 [0..getNumJoints(bodyUniqueId)]
可选用的	enableSensor	int	1/True 启用，0/False 禁用力/扭矩传感器
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

2.6 getLinkState(s)（获取连杆状态）

您还可以使用 getLinkState 查询每个链接的质心的笛卡尔世界位置和方向。 它还将质心的局部惯性框架报告给 URDF 链接框架，以便更容易计算图形/可视化框架。

getLinkState 输入参数

必要的	bodyUniqueId	int	loadURDF 等返回的正文唯一 ID
必要的	linkIndex	int	链接索引
可选用的	computeLinkVelocity	int	如果设置为 1，将计算并返回笛卡尔世界速度。
可选用的	computeForwardKinematics	int	如果设置为 1（或 True），笛卡尔世界位置/方向将使用正向运动学重新计算。
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

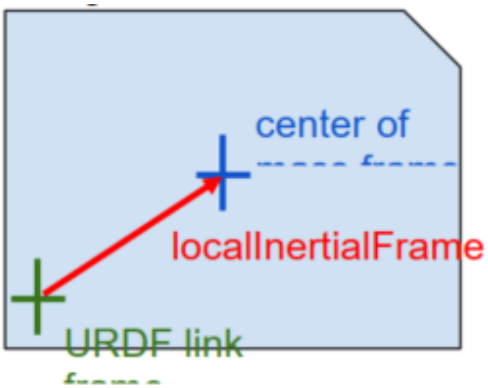
getLinkState 返回值

linkWorldPosition	vec3, list of 3 floats	质心的笛卡尔位置
linkWorldOrientation	vec4, list of 4 floats	质心的笛卡尔方向，四元数组[x,y,z,w]
localInertialFramePosition	vec3, list of 3 floats	以 URDF 链接框架表示的惯性框架（质心）的局部位置偏移
localInertialFrameOrientation	vec4, list of 4 floats	以 URDF 链接框架表示的惯性框架的局部方向（四元数 [x,y,z,w]）偏移。
worldLinkFramePosition	vec3, list of 3 floats	URDF 链接框架的世界位置
worldLinkFrameOrientation	vec4, list of 4 floats	URDF 链接框架的世界方向
worldLinkLinearVelocity	vec3, list of 3 floats	笛卡尔世界速度。 仅当 computeLinkVelocity 非零时才返回。
worldLinkAngularVelocity	vec3, list of 3 floats	笛卡尔世界速度。 仅当 computeLinkVelocity 非零时才返回。

URDF 链接框架和质心框架（都在世界空间中）之间的关系是：urdfLinkFrame = comLinkFrame * localInertialFrame.inverse()。 有关链接和惯性框架的更多信息，请参阅 ROS URDF tutorial.

2.6.1 getLinkStates（获取连杆状态）

getLinkStates 将返回多个链接的信息。 而不是 linkIndex，它将接受 linkIndices 作为 int 列表。 这可以通过减少多次调用 getLinkState 的调用开销来提高性能。



示例脚本（可能已过时，请检查实际的 Bullet/examples/pybullet/examples 文件夹。）

examples/pybullet/tensorflow/humanoid_running g.py	加载一个人形机器人并使用经过训练的神经网络来控制使用 TensorFlow 的运行，由 OpenAI 训练
examples/pybullet/gym/pybullet_envs/bullet/minitaur.py and minitaur_gym_env.py	OpenAI GYM 和 TensorFlow 的 Minitaur 环境 您也可以在执行 pip install pybullet 后使用 python -m pybullet_envs.examples.minitaur_gym_env_example 来查看 Minitaur 的运行情况。
examples/pybullet/examples/quadruped.py	从 URDF 文件加载四足动物，逐步模拟，根据正弦波控制电机以实现简单的跳跃步态。还将使用 p.startStateLogging 将状态记录到文件中。 见视频video.
examples/quadruped_playback.py	创建一个四足动物 (Minitaur)，读取日志文件并将位置设置为电机控制目标。
examples/pybullet/examples/testrender.py	加载 URDF 文件并渲染图像，获取像素（RGB、深度、分割掩码）并使用 MatPlotLib 显示图像。
examples/pybullet/examples/testrender_np.py	类似于 testrender.py，但使用 NumPy 数组加快像素传输。还包括简单的基准/计时。
examples/pybullet/examples/saveWorld.py	将对象的状态（位置、方向）保存到 pybullet Python 脚本中。 这主要用于在 VR 中设置场景并保存初始状态。 并非所有状态都被序列化。
examples/pybullet/examples/inverse_kinematic s.py	展示如何使用 calculateInverseKinematics 命令，创建一个 Kuka ARM 时钟
examples/pybullet/examples/rollPitchYaw.py	展示如何使用滑块 GUI 小部件
examples/pybullet/examples/constraint.py	以编程方式在链接之间创建约束。
examples/pybullet/examples/vrhand.py	使用由 VR 控制器跟踪的 VR 手套控制手。 见视频video.

2.7 getBaseVelocity, resetBaseVelocity（获取基座速度，重置本速度）

您可以使用 getBaseVelocity 访问物体底部的线速度和角速度。 输入参数为：

必要的	bodyUniqueId	int	body 唯一 ID，从 load* 方法返回。
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

这将返回两个 vector3 值的列表（列表中的 3 个浮点数），表示笛卡尔世界空间坐标中的线速度 [x,y,z] 和角速度 [wx,wy,wz]。

您可以使用 resetBaseVelocity 重置主体底部的线速度和/或角速度。 输入参数为：

必要的	objectUniqueId	int	body unique id, as returned from the load* methods.
可选用的	linearVelocity	vec3, list of 3 floats	linear velocity [x,y,z] in Cartesian world coordinates.
可选用的	angularVelocity	vec3, list of 3 floats	angular velocity [wx,wy,wz] in Cartesian world coordinates.
可选用的	physicsClientId	int	if you are connected to multiple servers, you can pick one.

2.8 applyExternalForce/Torque（施加外力/扭矩）

您可以使用 applyExternalForce 和 applyExternalTorque 对实体施加力或扭矩。 请注意，此方法仅在使用 stepSimulation 显式步进模拟时才有效，换句话说：setRealTimeSimulation(0)。 在每个模拟步骤之后，外力被清除为零。 如果您使用 'setRealTimeSimulation(1)，则 applyExternalForce/Torque 将具有未定义的行为（0、1 或多个力/扭矩应用程序）。

输入参数为：

必要的	objectUniqueId	int	加载方法返回的对象唯一 ID。
必要的	linkIndex	int	链接索引或基数为 -1。
必要的	forceObj	vec3, list of 3 floats	要应用的力/扭矩矢量 [x,y,z]。 请参阅坐标系的标志。
必要的	posObj	vec3, list of 3 floats	施加力的链接上的位置。 仅适用于 applyExternalForce。 请参阅坐标系的标志。
必要的	flags	int	指定力/位置的坐标系：WORLD_FRAME 用于笛卡尔世界坐标或 LINK_FRAME 用于局部链接坐标。
可选用的	physicsClientId	int	

2.9 getNumBodies, getBodyInfo, getBodyUniqueId, removeBody（获取Body数量，获取Body信息，获取Body特有的Id，删除 Body）

getNumBodies 将返回物理服务器中的物体总数。

如果您使用了“getNumBodies”，则可以使用“getBodyUniqueId”查询正文的唯一 ID。 请注意，所有 API 都已返回正文唯一 ID，因此如果您跟踪它们，通常不需要使用 getBodyUniqueId。

2.9.1 getBodyInfo （获取body的数据）

将返回从 URDF、SDF、MJCF 或其他文件中提取的基本名称。

2.9.2 syncBodyInfo

如果多个客户端连接到一个物理服务器改变世界（loadURDF、removeBody 等），syncBodyInfo 将同步身体信息（getBodyInfo）。

2.9.3 removeBody （移除body）

将通过其主体唯一 ID（来自 loadURDF、loadSDF 等）删除主体。

2.10 createConstraint, removeConstraint, changeConstraint（创建约束、移除约束、更改约束）

URDF、SDF 和 MJCF 将铰接体指定为无环的树结构。 “createConstraint”允许您连接主体的特定链接以关闭这些循环。 请参阅 Bullet/examples/pybullet/examples/quadruped.py 如何连接四足 5 杆闭环连杆的腿。 此外，您可以在对象之间以及对象与特定世界框架之间创建任意约束。 有关示例，请参见 Bullet/examples/pybullet/examples/constraint.py。

它还可用于控制由动画帧驱动的物理对象的运动，例如 VR 控制器。 最好使用约束，而不是直接为此目的设置位置或速度，因为这些约束与其他动力学约束一起求解。

createConstraint 具有以下输入参数：

必要的	parentBodyUniqueId	int	parent body unique id
必要的	parentLinkIndex	int	parent link index (or -1 for the base)
必要的	childBodyUniqueId	int	child body unique id, or -1 for no body (specify a non-dynamic child frame in world coordinates)
必要的	childLinkIndex	int	child link index, or -1 for the base
必要的	jointType	int	joint type: JOINT_PRISMATIC, JOINT_FIXED, JOINT_POINT2POINT, JOINT_GEAR
必要的	jointAxis	vec3, list of 3 floats	joint axis, in child link frame
必要的	parentFramePosition	vec3, list of 3 floats	position of the joint frame relative to parent center of mass frame.
必要的	childFramePosition	vec3, list of 3 floats	position of the joint frame relative to a given child center of mass frame (or world origin if no child specified)
可选用的	parentFrameOrientation	vec4, list of 4 floats	the orientation of the joint frame relative to parent center of mass coordinate frame
可选用的	childFrameOrientation	vec4, list of 4 floats	the orientation of the joint frame relative to the child center of mass coordinate frame (or world origin frame if no child specified)
可选用的	physicsClientId	int	if you are connected to multiple servers, you can pick one.

createConstraint 将返回一个整数唯一 id，可用于更改或删除约束。 有关 JOINT_GEAR 的示例，请参阅 examples/pybullet/examples/mimicJointConstraint.py 和 JOINT_POINT2POINT 的示例/pybullet/examples/minitaur.py 和 JOINT_FIXED 的示例/pybullet/examples/constraint.py。

2.10.1 changeConstraint（改变约束）

changeConstraint 允许您更改现有约束的参数。 输入参数为：

必要的	userConstraintUniqueId	int	createConstraint 返回的唯一 ID
可选用的	jointChildPivot	vec3, list of 3 floats	更新child pivot，请参阅“createConstraint”
可选用的	jointChildFrameOrientation	vec4, list of 4 floats	将子框架方向更新为四元数组
可选用的	maxForce	float	约束可以施加的最大力
可选用的	gearRatio	float	两个齿轮旋转速度之间的比率
可选用的	gearAuxLink	int	在某些情况下，例如差动驱动，第三个（辅助）链接用作参考位姿。 见 racecar_differential.py
可选用的	relativePositionTarget	float	两个齿轮之间的相对位置目标偏移量

可选用的	erp	float	约束误差减少参数
可选用的	physicsClientId	int	如果您连接到多台服务器，则可以选择一台.

另请参阅 Bullet/examples/pybullet/examples/constraint.py

removeConstraint 将删除由其唯一 ID 给出的约束。 其输入参数为：

必要的	userConstraintUniqueId	int	createConstraint 返回的唯一 ID
可选用的	physicsClientId	int	'connect' 返回的唯一 ID

2.11 getNumConstraints, getConstraintUniqueId（获取约束的数量，获取约束的唯一ID）

您可以查询使用 “createConstraint”创建的约束总数。 可选用的参数是 int PhysicsClientId。

2.11.1 getConstraintUniqueId（获取约束的唯一ID）

getConstraintUniqueId 将采用 0..getNumConstraints 范围内的序列索引，并报告约束唯一 ID。 请注意，约束唯一 ID 可能不连续，因为您可以删除约束。 输入是整数序列索引，并且任选使用一个物理客户端ID。

2.12 getConstraintInfo/State（获取约束信息/状态）

您可以查询约束信息，提供约束唯一 ID。 输入参数是

必要的	constraintUniqueId	int	createConstraint 返回的唯一 ID
可选用的	physicsClientId	int	'connect' 返回的唯一 ID

输出列表是：

parentBodyUniqueId	int	请参见 createConstraint
parentJointIndex	int	请参见 createConstraint
childBodyUniqueId	int	请参见 createConstraint
childLinkIndex	int	请参见 createConstraint
constraintType	int	请参见 createConstraint
jointAxis	vec3, list of 3 floats	请参见 createConstraint
jointPivotInParent	vec3, list of 3 floats	请参见 createConstraint
jointPivotInChild	vec3, list of 3 floats	请参见 createConstraint
jointFrameOrientationParent	vec4, list of 4 floats	请参见 createConstraint
jointFrameOrientationChild	vec4, list of 4 floats	请参见 createConstraint
maxAppliedForce	float	请参见 createConstraint
gearRatio	float	请参见 createConstraint

gearAuxLink	int	请参见 createConstraint
relativePositionTarget	float	请参见 createConstraint
erp	float	请参见 createConstraint

2.12.1 getConstraintState（获取约束状态）

给定一个唯一的约束 id，您可以在最近的模拟步骤中查询应用的约束力。 输入是一个约束唯一 id，输出是一个约束力向量，其维度是受约束影响的自由度（例如，固定约束影响 6 个自由度）。

2.13 getDynamicsInfo/changeDynamics（获取动力学信息/更改动力学参数）

您可以获得有关底座和连杆的质量、质心、摩擦力和其他属性的信息。

The input parameters to getDynamicsInfo are:

必要的	bodyUniqueId	int	对象唯一 ID，由 loadURDF 等返回。
必要的	linkIndex	int	连杆（关节）索引或 -1 为基础。
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

返回信息有限，我们会在需要时公开更多信息：

Mass（质量）	double	质量（以公斤为单位）
lateral_friction（横向摩擦系数）	double	摩擦系数
local inertia diagonal（局部惯性对角线）	vec3, list of 3 floats	局部惯性对角线。 请注意，连杆和底座以质心为中心，并与惯性主轴对齐。
local inertial pos（局部惯性位置）	vec3	惯性系在关节系局部坐标中的位置
local inertial orn（局部惯性方向）	vec4	惯性坐标系在关节坐标系局部坐标中的方向
Restitution（恢复系数）	double	恢复系数
rolling friction（滚动摩擦）	double	与接触法线正交的滚动摩擦系数
spinning friction（旋转摩擦）	double	围绕接触法线的旋转摩擦系数
contact damping（接触阻尼）	double	-1 如果不可用。 接触约束的阻尼。
contact stiffness（接触刚度）	double	-1 如果不可用。 接触约束的刚度。
body type	int	1 = 刚体，2 = 多体，3 = 软体
collision margin（碰撞余量）	double	高级/内部/不支持的信息。 碰撞形状的碰撞边缘。 碰撞边距取决于形状类型，它并不一致。

2.13.1 changeDynamics（更改动力学参数）

您可以使用 changeDynamics 更改质量、摩擦和恢复系数等属性。

输入参数为：

必要的	bodyUniqueId	int	对象唯一 ID，由 loadURDF 等返回。
必要的	linkIndex	int	连杆索引或基数为 -1
可选用的	mass	double	更改连杆的质量（或连杆索引 -1 的基数）
可选用的	lateralFriction	double	横向（线性）接触摩擦
可选用的	spinningFriction	double	接触法线周围的扭转摩擦
可选用的	rollingFriction	double	与接触法线正交的扭转摩擦（保持这个值非常接近零，否则模拟会变得非常不切实际。
可选用的	restitution	double	接触的弹性。 保持它略小于 1，最好接近 0。
可选用的	linearDamping	double	连杆的线性阻尼（默认为 0.04）
可选用的	angularDamping	double	连杆的角阻尼（默认为 0.04）
可选用的	contactStiffness	double	接触约束的刚度，与接触阻尼一起使用。
可选用的	contactDamping	double	此 body/link 的接触约束的阻尼。 与 contactStiffness 一起使用。 如果它是在联系部分的 URDF 文件中指定的，这将覆盖该值。
可选用的	frictionAnchor	int	启用或禁用摩擦锚(friction anchor)：摩擦漂移校正（默认禁用，除非在 URDF 接触部分中设置）
可选用的	localInertiaDiagnoal	vec3	惯性张量的对角元素。 请注意，底座和连杆以质心为中心并与惯性主轴对齐，因此惯性张量中没有非对角线元素。
可选用的	ccdSweptSphereRadiu	float	球体的半径来执行连续的碰撞检测。 有关示例，请参见 Bullet/examples/pybullet/examples/experimentalCcdSphereRadius.py。
可选用的	contactProcessingThreshold	float	距离低于此阈值的接触将由约束求解器处理。 例如，如果 contactProcessingThreshold = 0，则距离为 0.01 的联系人将不会作为约束处理。
可选用的	activationState	int	启用睡眠后，如果所有其他影响它的对象也准备好进入睡眠状态，则不移动（低于阈值）的对象将被禁用为睡眠状态。 pybullet.ACTIVATION_STATE_SLEEP pybullet.ACTIVATION_STATE_ENABLE_SLEEPING pybullet.ACTIVATION_STATE_DISABLE_WAKEUP 您还可以在 loadURDF 中使用 flags = pybullet.URDF_ENABLE_SLEEPING 来启用睡眠。 See sleeping.py example.

可选用的	jointDamping	double	应用于每个关节的关节阻尼系数。 该系数从 URDF 联合阻尼场中读取。 保持值接近 0。 （关节阻尼力）Joint damping force = -damping_coefficient * joint_velocity.
可选用的	anisotropicFriction	double	各向异性摩擦系数允许在不同方向上缩放摩擦。
可选用的	maxJointVelocity	double	给定关节的最大关节速度，如果在约束求解期间超过它，则将其固定。 默认最大关节速度为 100 个单位。
可选用的	collisionMargin	double	不受支持。 改变碰撞边缘。 取决于形状类型，它可能会或可能不会向碰撞形状添加一些填充。
可选用的	jointLowerLimit	double	改变关节的下限，也需要jointUpperLimit，否则会被忽略。 请注意，目前，“getJointInfo”中的关节限制未更新！
可选用的	jointUpperLimit	double	改变关节的上限，也需要jointLowerLimit，否则会被忽略。 请注意，目前，“getJointInfo”中的关节限制未更新！
可选用的	jointLimitForce	double	更改施加的最大力以满足关节限制。
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

2.14 setTimeStep（设置时间步长）

警告：在许多情况下，最好将 timeStep 保留为默认值，即 240Hz。 考虑到这个值，调整了几个参数。 例如，求解器迭代次数和接触、摩擦和非接触关节的误差减少参数 (erp) 与时间步长有关。 如果您更改时间步长，您可能需要相应地重新调整这些值，尤其是 erp 值。

您可以设置调用“stepSimulation”时使用的物理引擎时间步长。 最好只在模拟开始时调用此方法。 不要定期更改此时间步长。 setTimeStep 也可以使用新的 setPhysicsEngineParameter API 来实现。

输入参数为：

必要的	timeStep	float	每次调用“stepSimulation”时，timeStep 都会继续执行“timeStep”。
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

2.15 setPhysicsEngineParameter（设置物理引擎参数）

您可以使用 setPhysicsEngineParameter API 设置物理引擎参数。 公开了以下输入参数：

可选用的	fixedTimeStep	float	请参阅 setTimeStep 部分中的警告。 物理引擎时间步长（以秒为单位），每次调用“stepSimulation”模拟时间都会推进这个量。 与“setTimeStep”相同
可选用的	numSolverIterations	int	选择约束求解器迭代的最大次数。 如果达到solverResidualThreshold，则求解器可能会在numSolverIterations 之前终止。
可选用的	useSplitImpulse	int	高级功能，仅在使用最大坐标时：将位置约束求解和速度约束求解分为两个阶段，以防止巨大的穿透恢复力。
可选用的	splitImpulsePenetrationThreshold	float	与“useSplitImpulse”相关：如果特定接触约束的穿透小于此指定阈值，则该接触不会发生分裂脉冲。
可选用的	numSubSteps	int	通过“numSubSteps”进一步细分物理模拟步骤。 这将牺牲性能而不是准确性。
可选用的	collisionFilterMode	int	使用 0 作为默认碰撞过滤器：(group A&maskB) AND (groupB&maskA)。 使用 1 切换到 OR 碰撞过滤器：(group A&maskB) OR (groupB&maskA)
可选用的	contactBreakingThreshold	float	LCP 求解器不会处理距离超过此阈值的接触点。 此外，AABBs 也扩展了这个数字。 在 Bullet 2.x 中默认为 0.02。
可选用的	maxNumCmdPer1ms	int	实验性：如果执行的命令数超过此阈值，则添加 1ms 睡眠。
可选用的	enableFileCaching	int	设置为 0 以禁用文件缓存，例如 .obj 波前文件加载
可选用的	restitutionVelocityThreshold	float	如果相对速度低于此阈值，则恢复为零。
可选用的	erp	float	约束误差减少参数（非接触，非摩擦）

可选用的	contactERP	float	接触误差减少参数
可选用的	frictionERP	float	摩擦误差减少参数（启用位置摩擦锚时）
可选用的	enableConeFriction	int	设置为 0 以禁用隐式圆锥摩擦并使用金字塔近似（默认为圆锥）。 注意：虽然默认启用，但值得尝试禁用此功能，以防出现摩擦伪影。
可选用的	deterministicOverlappingPairs	int	设置为 1 以启用和 0 以禁用重叠对的排序（向后兼容性设置）。
可选用的	allowedCcdPenetration	float	如果启用连续碰撞检测 (CCD)，如果穿透低于此阈值，则不会使用 CCD。
可选用的	jointFeedbackMode	int	特定的联合反馈帧：JOINT_FEEDBACK_IN_WORLD_SPACE JOINT_FEEDBACK_IN_JOINT_FRAME
可选用的	solverResidualThreshold	double	速度阈值，如果每个约束的最大速度级别误差低于此阈值，则求解器将终止（除非求解器达到 numSolverIterations）。 默认值为 1e-7。
可选用的	contactSlop	float	低于此阈值时不解决接触的位置校正，以允许更稳定的接触。
可选用的	enableSAT	int	如果为true/1，则启用基于分离轴定理的凸碰撞检测，如果功能可用（而不是使用 GJK 和 EPA）。 在 loadURDF 中需要 URDF_INITIALIZE_SAT_FEATURES。 请参阅 satCollision.py 示例。
可选用的	constraintSolverType	int	实验性（最好忽略）：允许使用直接 LCP 求解器，例如 Dantzig。 请参阅 switchConstraintSolverType.py 示例。
可选用的	globalCFM	float	实验性（最好忽略）全局默认约束力混合参数。
可选用的	minimumSolverIslandSize	int	实验性（最好忽略），约束求解岛的最小尺寸，以避免非常小的独立约束岛。
可选用的	reportSolverAnalytics	int	当 true/1 时，额外的求解分析可用。
可选用的	warmStartingFactor	float	用于初始化初始求解器解决方案的前一帧力/脉冲的分数
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

2.15.1 setDefaultContactERP （设置默认接触参数）

setDefaultContactERP 是用于设置默认接触参数设置的 API。 它将融入 setPhysicsEngineParameter API。

2.16 getPhysicsEngineParameters（获取物理引擎参数）

您可以使用 getPhysicsEngineParameters 命令查询当前的一些物理引擎参数，使用任选使用的 'physicsClientId'。 这将返回参数的命名元组。

2.17 resetSimulation（重置仿真）

resetSimulation 将从世界中移除所有对象并将世界重置为初始条件。

可选用的	flags	int	实验性标志，最好忽略。 RESET_USE_SIMPLE_BROADPHASE RESET_USE_DEFORMABLE_WORLD RESET_USE_DISCRETE_DYNAMICS_WORLD
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

它需要一个任选使用的参数：物理客户端 ID（如果您创建了多个物理服务器连接）。

2.18 startStateLogging/stopStateLogging（记录开始状态/记录停止状态）

状态记录允许您记录模拟的状态，例如每个模拟步骤后一个或多个对象的状态（在每次调用 stepSimulation 之后或在启用 setRealTimeSimulation 时自动在每个模拟步骤之后）。这允许您记录对象的轨迹。 还可以选择记录身体的常见状态，例如基础位置和方向、关节位置（角度）和关节运动力。

使用 startStateLogging 生成的所有日志文件都可以使用 C++ 或 Python 脚本读取。 有关读取日志文件的 Python 脚本，请参阅 quadruped_playback.py 和 kuka_with_cube_playback.py。 您可以使用 bullet3/examples/Utils/RobotLoggingUtil.cpp/h 来读取 C++ 中的日志文件。

对于 MP4 视频录制，您可以使用日志记录选项 STATE_LOGGING_VIDEO_MP4。 我们计划实现各种其他类型的日志记录，包括记录 VR 控制器的状态。

作为一个特例，我们实现了 Minitaur 机器人的日志记录。 PyBullet 模拟的日志文件与真正的 Minitaur 四足日志文件相同。 有关示例，请参见 Bullet/examples/pybullet/examples/logMinitaur.py 。

重要提示：各种记录器都包含它们自己的内部时间戳，该时间戳在创建时从零开始。这意味着您需要同时启动所有记录器，以保持同步。 您需要确保模拟未在实时模式下运行，同时启动记录器：在创建记录器之前使用 pybullet.setRealTimeSimulation(0)。

必要的	loggingType	int	以下有各种类型的日志记录。 STATE_LOGGING_MINITAUR：这将需要从四足动物加载 quadruped/quadruped.urdf 和对象唯一 ID。 它记录时间戳、IMU 滚动/俯仰/偏航、8 个腿部电机位置 (q0-q7)、8 个腿部电机扭矩 (u0-u7)、躯干的前进速度和模式（在模拟中未使用）。 STATE_LOGGING_GENERIC_ROBOT：这将记录所有对象或选定对象的数据日志（如果提供了 objectUniqueIds）。 STATE_LOGGING_VIDEO_MP4：这将打开一个 MP4 文件并开始使用 ffmpeg 管道将 OpenGL 3D 可视化器像素流式传输到该文件。 它将需要安装 ffmpeg。 您也可以使用 avconv（Ubuntu 上的默认设置），只需创建一个符号链接，以便 ffmpeg 指向 avconv。 在 Windows 上，ffmpeg 存在一些问题，在某些情况下会导致撕裂/颜色伪影。 STATE_LOGGING_CONTACT_POINTS STATE_LOGGING_VR_CONTROLLES。 STATE_LOGGING_PROFILE_TIMINGS 这将以 JSON 格式转储一个计时文件，该文件可以使用 Google Chrome 打开 about://tracing LOAD。
必要的	fileName	string	存储日志文件数据的文件名（绝对或相对路径）。
可选用的	objectUniqueIds	list of int	如果留空，记录器可能会记录每个对象，否则记录器只会记录 objectUniqueIds 列表中的对象。
可选用的	maxLogDof	int	要记录的最大关节自由度数（不包括基本自由度）。 这适用于 STATE_LOGGING_GENERIC_ROBOT_DATA。 默认值为12. 如果机器人超过自由度数，它根本不会被记录。
可选用的	bodyUniqueIdA	int	适用于 STATE_LOGGING_CONTACT_POINTS。 如果提供，仅记录涉及 bodyUniqueIdA 的联系点。
可选用的	bodyUniqueIdB	int	适用于 STATE_LOGGING_CONTACT_POINTS。 如果提供，仅记录涉及 bodyUniqueIdB 的联系点。
可选用的	linkIndexA	int	适用于 STATE_LOGGING_CONTACT_POINTS。 如果提供，仅记录涉及 bodyUniqueIdA 的 linkIndexA 的联系点。
可选用的	linkIndexB	int	适用于 STATE_LOGGING_CONTACT_POINTS。 如果提供，仅记录涉及 bodyUniqueIdA 的 linkIndexB 的联系点。
可选用的	deviceTypeFilter	int	deviceTypeFilter 允许您选择要记录的 VR 设备：

			VR_DEVICE_CONTROLLER, VR_DEVICE_HMD ,VR_DEVICE_GENERIC_TRACKER 或任何它们的组合。 适用于 STATE_LOGGING_VR_CONTROLLES。默认值为 VR_DEVICE_CONTROLLER。
可选用的	logFlags	int	（即将推出的 PyBullet 1.3.1）。STATE_LOG_JOINT_TORQUES，记录由关节电机引起的关节扭矩。
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

该命令将返回一个非负 int loggingUniqueId，它可以与 stopStateLogging 一起使用。

Todo：记录为每种日志记录类型记录的数据。 现在，请使用日志读取实用程序查找或查看日志记录或 Python dumpLog.py 脚本的 C++ source code of the logging 。

2.18.1 stopStateLogging（停止状态记录）

您可以使用 loggingUniqueId 停止记录器。

2.18.2 submitProfileTiming（提交配置文件时间）

submitProfileTiming 允许插入开始和停止时间来分析 Python 代码。 请参阅 profileTiming.py示例。

PyBullet 和 Bullet 检测了许多功能，因此您可以查看时间花在了哪里。 您可以将这些配置文件时间转储到一个文件中，该文件可以使用 Google Chrome 在 about://tracing 窗口中使用 LOAD 功能进行查看。 在 GUI 中，您可以按“p”开始/停止配置文件转储。 在某些情况下，您可能想要检测客户端代码的计时。 您可以使用 PyBullet 提交配置文件时间。 这是一个示例输出：

3. Deformables and Cloth (FEM, PBD)（变形体与表面体（有限元，动力学））

除了铰接多体和刚体动力学，PyBullet 还使用有限元法 (FEM) 质量弹簧系统以及基于位置的动力学 (PBD) 实现可变形和布料模拟。 使用该实现的一些研究论文是学习使用目标条件运输网络 Learning to Rearrange Deformable Cables, Fabrics, and Bags with Goal-Conditioned Transporter Networks （使用 FEM，ICRA2021）重新排列可变形电缆、织物和袋子、用于可变形对象操作的 Sim-to-Real 强化学习Sim-to-Real Reinforcement Learning for Deformable Object Manipulation （使用 PBD）和辅助Gym：一种辅助机器人物理模拟框架Assistive Gym: A Physics Simulation Framework for Assistive Robotics。 （使用 PBD）。

PyBullet 实现了多体/刚体和可变形体之间的双向耦合。 双向耦合适用于碰撞以及手动创建的附件（请参阅下面的 createSoftBodyAnchor）。

默认情况下，PyBullet 将使用基于位置的动态 (PBD)。 您可以通过如下重置世界来启用基于有限元方法 (FEM) 的模拟：

pybullet.resetSimulation(p.RESET_USE_DEFORMABLE_WORLD) 另请参见deformable_torus.py 作为示例。

3.1 loadSoftBody/loadURDF（加载软体/加载 URDF）

有几种方法可以创建可变形对象，包括布料或体积。 loadSoftBody 允许您从 VTK 或 OBJ 文件加载可变形对象。

以下参数可用于 loadSoftBody。 请注意，有几个参数假设您使用 FEM 模型并且对 PBD 仿真没有影响。

必要的	fileName	string	可变形体的文件名。 可以是 Wavefront .obj 格式或 VTK 格式。
可选用的	basePosition	vec3	可变形物体的初始位置
可选用的	baseOrientation	vec4	可变形对象的初始方向（四元数 x,y,z,w）
可选用的	scale	double	调整可变形对象大小的缩放因子（默认值 = 1）
可选用的	mass	double	可变形物体的总质量，质量均匀分布在所有顶点之间
可选用的	collisionMargin	double	碰撞边缘扩展了可变形对象，它可以帮助避免穿透，尤其是对于薄（布）可变形对象
可选用的	useMassSpring	bool	使用质量弹簧
可选用的	useBendingSprings	bool	创建弯曲弹簧以控制可变形对象的弯曲
可选用的	useNeoHookean	bool	启用 Neo Hookean 模拟
可选用的	springElasticStiffness	double	刚度参数
可选用的	springDampingStiffness	double	阻尼参数
可选用的	springDampingAllDirections	double	弹簧阻尼参数

可选用的	springBendingStiffness	double	弯曲刚度参数
可选用的	NeoHookeanMu	double	Neo Hookean 模型的参数
可选用的	NeoHookeanLambda	double	Neo Hookean 模型的参数
可选用的	NeoHookeanDamping	double	Neo Hookean 模型的参数
可选用的	frictionCoeff	double	可变形物体的接触摩擦
可选用的	useFaceContact	bool	启用面内部的碰撞，而不仅仅是顶点。
可选用的	useSelfCollision	bool	为可变形对象启用自碰撞
可选用的	repulsionStiffness	double	有助于避免穿透的参数。

3.1.1 loadURDF（加载 URDF）

PyBullet 还扩展了 URDF 格式以使用“可变形”标签指定可变形对象。 参见例如 deformable_torus.urdf :

```
<robot name="torus">
 <deformable name="torus">
   <inertial>
     <mass value="1" />
     <inertia ixx="0.0" ixy="0" ixz="0" iyy="0" iyz="0" izz="0" />
   </inertial>
   <collision_margin value="0.006"/>
   <repulsion_stiffness value="800.0"/>
   <friction value= "0.5"/>
   <neohookean mu= "60" lambda= "200" damping= "0.01" />
   <visual filename="torus.vtk"/>
 </deformable>
</robot>
```

3.2 createSoftBodyAnchor（创建软体锚）

您可以将可变形对象的顶点固定到世界，或使用 createSoftBodyAnchor 将可变形对象的顶点附加到多体。 这将返回一个约束唯一 ID。 您可以使用 removeConstraint API 删除此约束。

有关示例，请参见 deformable_anchor.py 。

您可以使用 getMeshData API 访问可变形对象的顶点。

4. Synthetic Camera Rendering（合成相机渲染）

PyBullet 有一个内置的 OpenGL GPU 可视化器和一个基于 TinyRenderer 的内置 CPU 渲染器。 这使得从任意相机位置渲染图像变得非常容易。 在 Linux 上，您还可以在没有 X11 上下文的情况下启用硬件加速的 OpenGL 渲染，例如在 Google Cloud Platform 或 Colab 中进行云渲染。 请参阅eglRenderTest.py 示例如何使用它，如“插件”部分所述。

合成相机由两个 4 x 4 矩阵指定：视图矩阵和投影矩阵。 由于这些不是很直观，因此有一些辅助方法可以根据可理解的参数计算视图和投影矩阵。

查看这篇this article 关于内在相机矩阵的文章，其中包含指向 OpenGL 相机信息的链接。

4.1 computeView/ProjectionMatrix（计算机视图/投影矩阵）

computeViewMatrix 输入参数是

必要的	cameraEyePosition	vec3, list of 3 floats	笛卡尔世界坐标中的眼睛位置
必要的	cameraTargetPosition	vec3, list of 3 floats	目标（焦点）点的位置，以笛卡尔世界坐标表示
必要的	cameraUpVector	vec3, list of 3 floats	相机的向上向量，在笛卡尔世界坐标中
可选用的	physicsClientId	int	未使用，为 API 一致性而添加

输出是 4x4 视图矩阵，存储为 16 个浮点数的列表。

4.1.1 computeViewMatrixFromYawPitchRoll（从 Yaw Pitch Roll 三轴计算视图矩阵）

输入参数是

必要的	cameraTargetPosition	list of 3 floats	笛卡尔世界坐标中的目标焦点
必要的	distance	float	眼睛到焦点的距离
必要的	yaw	float	绕上轴的左/右偏航角。
必要的	pitch	float	以度数向上/向下倾斜。
必要的	roll	float	围绕前向矢量滚动度数
必要的	upAxisIndex	int	1 表示 Y 或 2 表示 Z 轴向上。
可选用的	physicsClientId	int	未使用，为 API 一致性而添加。

输出是 4x4 视图矩阵，存储为 16 个浮点数的列表。

4.1.2 computeProjectionMatrix（计算投影矩阵）

输入参数是

必要的	left	float	左屏（画布）坐标
必要的	right	float	右屏幕（画布）坐标
必要的	bottom	float	底部屏幕（画布）坐标
必要的	top	float	顶部屏幕（画布）坐标
必要的	near	float	近平面距离
必要的	far	float	远平面距离
可选用的	physicsClientId	int	未使用，为 API 一致性而添加。

输出是 4x4 投影矩阵，存储为 16 个浮点数的列表。

4.1.3 computeProjectionMatrixFOV（计算投影矩阵FOV）

此命令还将使用不同的参数返回一个 4x4 投影矩阵。 您可以查看 OpenGL 文档以了解参数的含义。

输入参数为：

必要的	fov	float	视野
必要的	aspect	float	纵横比
必要的	nearVal	float	近平面距离
必要的	farVal	float	远平面距离
可选用的	physicsClientId	int	未使用，为 API 一致性而添加。

4.2 getCameraImage（获取相机图像）

getCameraImage API 将返回一个 RGB 图像、一个深度缓冲区和一个分段掩码缓冲区，其中每个像素的可见对象的主体唯一 ID。 请注意，可以使用 numpy 选项编译 PyBullet：使用 numpy 将提高将相机像素从 C 复制到 Python 的性能。 注意：旧的 renderImage API 已过时并被 getCameraImage 取代。

getCameraImage 输入参数：

必要的	width	int	以像素为单位的水平图像分辨率
必要的	height	int	以像素为单位的垂直图像分辨率
可选用的	viewMatrix	16 floats	4x4 视图矩阵，请参阅 computeViewMatrix*
可选用的	projectionMatrix	16 floats	4x4 投影矩阵，请参阅 computeProjection*

可选用的	lightDirection	vec3, list of 3 floats	lightDirection 指定光源的世界位置，方向是从光源位置到世界框架的原点。
可选用的	lightColor	vec3, list of 3 floats	[RED,GREEN,BLUE] 0..1 范围内的定向光颜色，仅适用于 ER_TINY_RENDERER
可选用的	lightDistance	float	光沿标准化 lightDirection 的距离，仅适用于 ER_TINY_RENDERER
可选用的	shadow	int	1 表示阴影，0 表示没有阴影，仅适用于 ER_TINY_RENDERER
可选用的	lightAmbientCoeff	float	光照环境系数，仅适用于 ER_TINY_RENDERER
可选用的	lightDiffuseCoeff	float	光漫反射系数，仅适用于 ER_TINY_RENDERER
可选用的	lightSpecularCoeff	float	光镜面系数，仅适用于 ER_TINY_RENDERER
可选用的	renderer	int	ER_BULLET_HARDWARE_OPENGL 或 ER_TINY_RENDERER。请注意，DIRECT 模式没有 OpenGL，因此需要 ER_TINY_RENDERER。
可选用的	flags	int	ER_SEGMENTATION_MASK_OBJECT_AND_LINKINDEX，见下文segmentationMaskBuffer 和示例代码的描述。使用 ER_NO_SEGMENTATION_MASK 来避免计算分段掩码。
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

getCameraImage 返回参数列表：

width	int	宽度 图像分辨率（以像素为单位）（水平）
height	int	以像素为单位的高度图像分辨率（垂直）

rgbPixels	list of [char RED,char GREEN,char BLUE, char ALPHA] [0..width*height]	R,G,B,A 格式的像素颜色列表，每种颜色的范围为 [0..255]
depthPixels	list of float [0..width*height]	R、G、B、A 格式的像素颜色列表，范围为 [0..255]，用于每个颜色深度缓冲区。 Bullet 使用 OpenGL 进行渲染，约定是非线性 z-buffer。 请参阅 https://stackoverflow.com/questions/6652253/getting-the-true-z-value-from-the-depth-buffer far=1000.//取决于投影矩阵，这是默认的near=0.01//取决于投影矩阵 depth = far * near / (far - (far - near) * depthImg)//depthImg 是来自 Bullet 'getCameraImage' 的深度 另请参阅 https://github.com/bulletphysics/bullet3/blob/master/examples/pybullet/examples/pointCloudFromCameraImage.py
segmentationMaskBuffer	list of int [0..width*height]	对于每个像素，可见对象的唯一 ID。如果 ER_SEGMENTATION_MASK_OBJE 使用 CT_AND_LINKINDEX，segmentationMaskBuffer 结合对象唯一id和链接索引如下： value = objectUniqueId + (linkIndex+1)<<24. See example. 所以对于一个没有关节/链接的自由浮动体，分割掩码等于它的body唯一ID，因为它的链接索引是-1。

4.2.1 isNumpyEnabled（是否启用了 Numpy）

请注意，对于大图像，将像素从 C/C++ 复制到 Python 可能非常慢，除非您使用 NumPy 编译 PyBullet。 您可以使用 PyBullet.isNumpyEnabled() 检查是否启用了 NumPy。 pip install pybullet 已启用 NumPy（如果系统上可用）。 目前，只有 getCameraImage 使用 numpy 加速。

4.3 getVisualShapeData（获取视觉形状数据）

您可以使用 getVisualShapeData 访问视觉形状信息。 您可以使用它来将您自己的渲染方法与 PyBullet 模拟连接起来，并在每个模拟步骤之后手动同步世界变换。 您还可以使用 getMeshData（尤其是对于可变形对象）来接收有关顶点位置的数据。

输入参数为：

必要的	objectUniqueId	int	对象唯一 ID，由加载方法返回。
可选用的	flags	int	VISUAL_SHAPE_DATA_TEXTURE_UNIQUE_IDS 也会提供 textureUniqueId
可选用的	physicsClientId	int	'connect' 返回的物理客户端 ID

输出是一个视觉形状数据列表，每个视觉形状的格式如下：

objectUniqueId	int	对象唯一ID，与输入相同
linkIndex	int	连杆索引或基座为 -1
visualGeometryType	int	视觉几何类型（待定）
dimensions	vec3, list of 3 floats	几何尺寸（大小、局部比例）
meshAssetFileName	string, list of chars	三角形网格的路径（如果有）。 通常相对于 URDF、SDF 或 MJCF 文件位置，但也可以是绝对的。
localVisualFrame position	vec3, list of 3 floats	局部视觉框架的位置，相对于链接/关节框架
localVisualFrame orientation	vec4, list of 4 floats	局部视觉框架相对于链接/关节框架的方向
rgbaColor	vec4, list of 4 floats	以红色/绿色/蓝色/alpha 表示的 URDF 颜色（如果指定）
textureUniqueId	int	（该字段仅在使用 VISUAL_SHAPE_DATA_TEXTURE_UNIQUE_IDS 标志时才存在）纹理形状的唯一 ID，如果没有则为 -1

物理模拟在 getBasePositionAndOrientation 和 getLinkState 中使用质心作为笛卡尔世界变换的参考。 如果您实现自己的渲染，则需要利用质心世界变换和（反向）localInertialFrame 将局部视觉变换转换为世界空间。 您可以使用 getLinkState API 访问 localInertialFrame。

4.4 changeVisualShape, loadTexture（改变视觉形状，加载纹理）

您可以使用 changeVisualShape 更改形状的纹理、RGBA 颜色和其他属性。

必要的	objectUniqueId	int	对象唯一 ID，由 load 方法返回。
必要的	linkIndex	int	链接索引
可选用的	shapeIndex	int	内部使用，建议忽略 shapeIndex 或将其保留为 -1。 目的是让您选择一个特定的形状索引来修改，因为 URDF（和 SDF 等）每个链接可以有超过 1 个视觉形状。 此 shapeIndex 与 getVisualShapeData 返回的列表排序相匹配。

可选用的	textureUniqueId	int	纹理唯一 ID，由“loadTexture”方法返回
可选用的	rgbaColor	vec4, list of 4 floats	RED、GREEN、BLUE 和 ALPHA 的颜色分量，每个都在 [0..1] 范围内。 Alpha 目前必须为 0（不可见）或 1（可见）。 请注意，TinyRenderer 不支持透明度，但 GUI/EGL OpenGL3 渲染器支持。
可选用的	specularColor	vec3	镜面反射颜色分量，RED、GREEN 和 BLUE，可以从 0 到大数 (>100)。
必要的	physicsClientId	int	'connect' 返回的物理客户端 ID

4.4.1 loadTexture

如果加载成功，则从文件加载纹理并返回非负纹理唯一 ID。 这个唯一的 id 可以与 changeVisualShape 一起使用。

5. Collision Detection Queries（碰撞检测查询）

您可以查询上一个“stepSimulation”过程中存在的联系点信息。 要获取联系点，您可以使用“getContactPoints”API。 请注意，“getContactPoints”不会重新计算任何联系点信息。

5.1 getOverlappingObjects, getAABB（获取重叠对象，获取AABB）

此查询将返回具有轴对齐边界框与给定轴对齐边界框重叠的对象的所有唯一 ID。 请注意，查询是保守的，可能会返回没有实际 AABB 重叠的其他对象。 发生这种情况是因为加速度结构具有一些启发式方法，会稍微扩大 AABB（额外的余量并沿速度矢量挤压）。

getOverlappingObjects 输入参数为：

必要的	aabbMin	vec3, list of 3 floats	aabb 的最小坐标
必要的	aabbMax	vec3, list of 3 floats	aabb的最大坐标
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

getOverlappingObjects 将返回对象唯一 ID 的列表。

5.1.1 getAABB

您可以在给定对象唯一 ID 的情况下查询轴对齐的边界框（在世界空间中），并且可以选择使用链接索引。 （当你不通过链接索引，或者使用-1，你得到基地的AABB）.

输入参数是：

必要的	bodyUniqueId	int	创建方法返回的对象唯一 ID。
可选用的	linkIndex	int	范围 [0..getNumJoints(..)] 中的链接索引
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

返回结构是世界空间坐标中 vec3、aabbMin (x,y,z) 和 aabbMax (x,y,z) 的列表。

另请参见getAABB.py 示例。

5.2 getContactPoints, getClosestPoints（获取接触点，获取最近点）

getContactPoints API 返回在最近一次调用 stepSimulation 期间计算的接触点。 请注意，如果您在 stepSimulation 之后更改模拟状态，则 “getContactPoints”不会更新并且可能无效。 其输入参数如下：

可选用的	bodyA	int	只报告涉及body A 的接触点
可选用的	bodyB	int	只报告涉及body B的接触点。重要提示：如果你提供body B，你需要有一个有效的bodyA。
可选用的	linkIndexA	int	只报告涉及bodyA的linkIndexA的接触点
可选用的	linkIndexB	int	只报告涉及bodyB的linkIndexB的接触点
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

getContactPoints 将返回联系点列表。 每个联络点都有以下字段：

contactFlag	int	预订的
bodyUniqueIdA	int	body A唯一的 body id

bodyUniqueIdB	int	body B唯一的 bodyID
linkIndexA	int	主体 A 的链接索引， -1表示基座
linkIndexB	int	主体 B 的链接索引，-1 表示基座
positionOnA	vec3, list of 3 floats	A 上的接触位置，在笛卡尔世界坐标中
positionOnB	vec3, list of 3 floats	B 上的接触位置，在笛卡尔世界坐标中
contactNormalOnB	vec3, list of 3 floats	在 B 上接触法线，指向 A
contactDistance	float	接触距离，分离为正，穿透为负
normalForce	float	在最后一个“stepSimulation”中施加的法向力
lateralFriction1	float	lateralFrictionDir1 方向的横向摩擦力
lateralFrictionDir1	vec3, list of 3 floats	第一横向摩擦方向
lateralFriction2	float	lateralFrictionDir2 方向的横向摩擦力
lateralFrictionDir2	vec3, list of 3 floats	第二横向摩擦方向

5.2.1 getClosestPoints（获取最近点）

也可以独立于 stepSimulation 计算最近点。 这还允许您计算具有任意分离距离的对象的最近点。 在此查询中，不会报告法向力。

getClosestPoints 输入参数：

必要的	bodyA	int	第一个对象 (A) 的对象唯一 ID
必要的	bodyB	int	第二个对象的对象唯一 ID (B)
必要的	distance	float	如果对象之间的距离超过此最大距离，则可能不会返回任何点。
可选用的	linkIndexA	int	对象 A 的链接索引（基座为 -1）
可选用的	linkIndexB	int	对象 B 的链接索引（基座为 -1）
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

getClosestPoints 返回与 getContactPoints 格式相同的最近点列表（但在这种情况下 normalForce 始终为零）

5.3 rayTest, rayTestBatch（射线测试）

您可以执行单次光线投射来查找第一个命中对象的交集信息。 rayTest 输入参数是:

必要的	rayFromPosition	vec3, list of 3 floats	世界坐标中射线的起点
必要的	rayToPosition	vec3, list of 3 floats	世界坐标中射线的终点
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

raytest 查询将在相交的情况下返回以下信息：

objectUniqueId	int	命中对象的对象唯一 ID
linkIndex	int	命中对象的链接索引，如果没有/父对象，则为 -1。
hit fraction	float	沿射线在 [0,1] 范围内命中分数。
hit position	vec3, list of 3 floats	笛卡尔世界坐标中的命中位置
hit normal	vec3, list of 3 floats	在笛卡尔世界坐标中正常命中

5.3.1 rayTestBatch（射线测试批次）

这与 rayTest 类似，但允许您提供光线数组，以便更快地执行。 “rayFromPositions”的大小需要等于“rayToPositions”的大小。 即使没有相交，您也可以每条射线产生一个射线结果：您需要使用 objectUniqueId 字段来检查射线是否击中任何东西：如果 objectUniqueId 为 -1，则没有击中。 在这种情况下，“命中分数”为 1。每批次的最大光线数为 pybullet.MAX_RAY_INTERSECTION_BATCH_SIZE。

rayTestBatch 输入参数为：

必要的	rayFromPositions	list of vec3, list of list of 3 floats	每条射线的起点列表，以世界坐标表示
必要的	rayToPositions	list of vec3, list of list of 3 floats	世界坐标中每条射线的端点列表
可选用的	parentObjectUniqueId	int	来自/到的光线在父对象的本地空间中
可选用的	parentLinkIndex	int	来自/到的光线在父对象的本地空间中
可选用的	numThreads	int	使用多个线程来计算射线测试（0 = 使用所有可用线程，正数 = 正好这个线程数，默认 = -1 = 单线程）
可选用的	reportHitNumber	int	而不是第一个最近的命中，您可以报告第 n 个命中

可选用的	collisionFilterMask	int	而不是第一个最接近的命中，只有在collisionFilterMask和body碰撞过滤器组之间的按位和非零时才测试命中。 见setCollisionFilterGroupMask 如何修改 body filter mask/group.you可以报告第n次命中
可选用的	fractionEpsilon	float	仅在使用reportHitNumber 时有用：如果分数与该分数Epsilon 中的现有命中相似，则在命中同一主体时忽略重复命中。 例如，一条射线可能会击中一个物体的许多共面三角形，您可能只对其中一个感兴趣。
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

输出是每条输入射线的一个射线相交结果，其信息与上述 rayTest 查询中的信息相同。 请参阅 batchRayTest.py 示例如何使用它。

5.4 getCollisionShapeData（获取碰撞形状数据）

您可以使用该查询查询现有体基和链接的碰撞几何类型和其他碰撞形状信息。 它的工作原理与 getVisualShapeData 非常相似。

getCollisionShapeData 的输入参数为：

必要的	objectUniqueId	int	对象唯一 ID，从 loadURDF 等接收
必要的	linkIndex	int	连杆索引或基座为 -1
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

返回值是一个包含以下内容的列表：

object unique id	int	对象唯一标识
linkIndex	int	连杆索引或基座为 -1
geometry type	int	几何类型： GEOM_BOX 、 GEOM_SPHERE 、 GEOM_CAPSULE 、 GEOM_MESH、GEOM_PLANE
dimensions	vec3	取决于几何类型：对于 GEOM_BOX：范围，对于 GEOM_SPHERE 尺寸 [0] = 半径，对于 GEOM_CAPSULE 和 GEOM_CYLINDER，尺寸 [0] = 高度（长度），尺寸 [1] = 半径。 对于 GEOM_MESH，尺寸是比例因子。
filename	string	仅适用于 GEOM_MESH：碰撞网格资源的文件名（和路径）
local frame pos	vec3	碰撞框架相对于质心/惯性框架的局部位置。
local frame orn	vec4	碰撞框架相对于惯性框架的局部方向。

5.5 Enable/Disable Collisions（启用/禁用碰撞）

默认情况下，不同动态运动体之间的碰撞检测是启用的。 可以使用诸如 loadURDF 中的 'URDF_USE_SELF_COLLISION' 标志之类的标志来启用同一主体的链接之间的自碰撞（有关详细信息，请参阅 loadURDF 命令）。

您可以使用 setCollisionFilterGroupMask API 启用和禁用对象组之间的碰撞检测。

5.5.1 V-HACD（体积分层近似分解）

PyBullet 包括 Khaled Mamou 的 Volumetric Hierarchical Approximate Decomposition (V-HACD) 的实现。 这可以导入一个凹波前 .obj 文件并导出一个包含凸分解部分的新波前 obj 文件。 这可以在 PyBullet 中用于有效地处理凹形移动几何体。

对于静态（非移动）凹三角形网格环境，您可以使用 URDF 文件中的标签 (<link concave="yes" name="baseLink">) 或使用带有 flags=p.GEOM_FORCE_CONCAVE_TRIMESH 的 createCollisionShape 将三角形网格标记为凹面。

必要的	fileNameIn	string	源（凹）Wavefront obj obj 文件名
必要的	fileNameOut	string	目的地（凸分解） Wavefront obj 文件名
必要的	fileNameLog	string	日志文件名
可选用的	concavity	double	最大允许凹度（默认=0.0025，范围=0.0-1.0）
可选用的	alpha	double	控制沿对称平面剪裁的偏差（默认=0.05，范围=0.0-1.0）
可选用的	beta	double	控制沿旋转轴剪裁的偏差（默认值=0.05，范围=0.0-1.0）
可选用的	gamma	double	控制合并阶段允许的最大凹度（默认=0.00125，范围=0.0-1.0）
可选用的	minVolumePerCH	double	控制生成的凸包的自适应采样（默认=0.0001，范围=0.0-0.01）
可选用的	resolution	int	体素化阶段生成的最大体素数（默认=100,000，范围=10,000-16,000,000）
可选用的	maxNumVerticesPerCH	int	控制每个凸包的最大三角形数（默认=64，范围=4-1024）
可选用的	depth	int	剪辑阶段的最大数量。 在每个分割阶段，凹度高于用户定义阈值的部分将根据最佳剪切平面进行剪切（默认=20，范围=1-32）
可选用的	planeDownsampling	int	控制搜索“最佳”剪切平面的粒度（默认=4，范围=1-16）
可选用的	convexhullDownsampling	int	在裁剪平面选择阶段控制凸包生成过程的精度（默认=4，范围=1-16）
可选用的	pca	int	在应用凸分解之前启用/禁用归一化网格（默认值=0，范围={0,1}）
可选用的	mode	int	0：基于体素的近似凸分解，1：基于四面体的近似凸分解（默认=0，范围={0,1}）

可选用的	convexhullApproximation	int	计算凸包时启用/禁用近似（默认=1，范围={0,1}）
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。 注意：vhacd 分解目前发生在客户端。

Example usage:

```
import pybullet as  p
import pybullet_data as pd
import os
p.connect(p.DIRECT)
name_in      =      os.path.join(pd.getDataPath(),
"duck.obj") name_out = "duck_vhacd2.obj"
name_log = "log.txt"
p.vhacd(name_in, name_out, name_log)
```

5.5.2 setCollisionFilterGroupMask（设置碰撞过滤器组掩码）

每个身体都是一个群体的一部分。 如果它们的组与掩码匹配，它会与其他物体发生碰撞，反之亦然。 使用涉及的两个主体的组和掩码执行以下检查。 这取决于碰撞过滤器模式。

必要的	bodyUniqueId	int	bodyUniqueId of the body to be configured
必要的	linkIndexA	int	link index of the body to be configured
必要的	collisionFilterGroup	int	bitwise group of the filter, see below for explanation
必要的	collisionFilterMask	int	bitwise mask of the filter, see below for explanation
可选用的	physicsClientId	int	if you are connected to multiple servers, you can pick one.

您可以对特定链接对之间的碰撞检测进行更细粒度的控制。 在那里使用 setCollisionFilterPair API：您可以启用或禁用碰撞检测。 setCollisionFilterPair 将覆盖过滤器组/掩码和其他逻辑。

5.5.3 setCollisionFilterPair（设置碰撞过滤器）

必要的	bodyUniqueIdA	int	待过滤的body A 的唯一的body ID
必要的	bodyUniqueIdB	int	待过滤的body B的唯一的body Id，A==B表示自碰撞
必要的	linkIndexA	int	主体 A 的链接索引
必要的	linkIndexB	int	主体 B 的链接索引
必要的	enableCollision	int	1 启用碰撞，0 禁用碰撞
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

还有一个插件 API 可以编写您自己的碰撞过滤实现，请参阅 collisionFilterPlugin implementation。

6. Inverse Dynamics, Kinematics（逆动力学、正运动学）

6.1 calculateInverseDynamics(2)（计算逆动力学）

calculateInverseDynamics 将从指定的关节位置和速度开始计算达到给定关节加速度所需的力。 使用递归牛顿欧拉算法 (RNEA) 计算逆动力学。

calculateInverseDynamics 输入参数为：

必要的	bodyUniqueId	int	body 唯一 ID，由 loadURDF 等返回。
必要的	objPositions	list of float	每个自由度 (DoF) 的关节位置（角度）。 请注意，固定关节的自由度为 0。 在所有情况下（浮动底座和固定底座）都会跳过/忽略底座。
必要的	objVelocities	list of float	每个自由度 (DoF) 的关节速度
必要的	objAccelerations	list of float	每个自由度 (DoF) 的期望关节加速度
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

calculateInverseDynamics 返回每个自由度的关节力列表。

请注意，当涉及多自由度（球形）关节时，calculateInverseDynamics 使用不同的代码路径并且速度稍慢。 另请注意，calculateInverseDynamics 忽略关节/连杆阻尼，而正向动力学（在 stepSimulation 中）包括那些阻尼项。 因此，如果您想比较反向动力学和正向动力学，请确保使用 changeDynamics 和 jointDamping 将这些阻尼项设置为零，并通过 linearDamping 和 angularDamping 进行链接阻尼。

6.2 calculateJacobian, MassMatrix（计算雅可比，质量矩阵）

calculateJacobian 将计算链接上一个点的平移和旋转雅可比，

例如 x_dot = J * q_dot。 根据根链接是固定的还是浮动的，返回的雅可比值略有不同。 如果浮动，雅可比将包括对应于根链接自由度的列； 如果固定，雅可比将只有与关节相关的列。函数调用获取运动学状态的完整描述，这是因为实际首先调用了calculateInverseDynamics，并从中提取了所需的雅可比； 因此，如果需要，为关节速度和加速度传递零向量是合理的。

calculateJacobian 输入参数为：

必要的	bodyUniqueId	int	body 唯一 ID，由 loadURDF 等返回。
必要的	linkIndex	int	雅可比的连杆索引。
必要的	localPosition	list of float	指定连杆上要计算雅可比的点，在其质心周围的连杆本地坐标中。
必要的	objPositions	list of float	关节位置（角度）
必要的	objVelocities	list of float	关节速度
必要的	objAccelerations	list of float	期望的关节加速度
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

calculateJacobian returns:

必要的	linearJacobian	mat3x ((dof), (dof), (dof))	平移雅可比，x_dot = J_t * q_dot。
必要的	angularJacobian	mat3x ((dof), (dof), (dof))	旋转雅可比，r_dot = J_r * q_dot。

6.2.1 calculateMassMatrix（计算质量矩阵）

calculateMassMatrix 将根据关节位置计算关节体的系统惯性。 复合刚体算法 (CBRA) 用于计算质量矩阵。

必要的	bodyUniqueId	int	body 唯一 ID，由 loadURDF 等返回。
必要的	objPositions	array of float	每个连杆的关节位置。
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

结果是尺寸为 dofCount * dofCount 的方形质量矩阵，存储为 dofCount 行的列表，每一行都是 dofCount 质量矩阵元素的列表。

请注意，当涉及多自由度（球形）关节时，calculateMassMatrix 将使用不同的代码路径，这会慢一些。

6.3 Inverse Kinematics（逆运动学）

您可以计算使末端执行器到达笛卡尔世界空间中给定目标位置的关节角度。 在内部，Bullet 使用改进版的 Samuel Buss Inverse Kinematics 库。 目前仅公开了带有或不带有 Null Space 控件的阻尼最小二乘法，具有单个末端执行器目标。 您也可以随意指定末端执行器的目标方向。此外，还有一个选项可以使用零空间来指定关节限制和静止姿势。 这种任选使用的空空间支持需要所有 4 个列表（lowerLimits、upperLimits、jointRanges、restPoses），否则将使用常规 IK。有关详细信息，另请参见 Bullet/examples/pybullet/examples 文件夹中的 inverse_kinematics.py 示例。

6.3.1 calculateInverseKinematics(2)

calculateInverseKinematics 输入参数为：

必要的	bodyUniqueId	int	正文唯一 ID，由 loadURDF 返回
必要的	endEffectorLinkIndex	int	末端执行器链接索引
必要的	targetPosition	vec3, list of 3 floats	末端执行器的目标位置（它的连杆坐标，而不是质心坐标！）。 默认情况下，这是在笛卡尔世界空间中，除非您提供 currentPosition 关节角度。
可选用的	targetOrientation	vec3, list of 4 floats	笛卡尔世界空间中的目标方向，四元数 [x,y,w,z]。 如果未指定，将使用纯位置 IK。
可选用的	lowerLimits	list of floats [0..nDof]	任选使用空空间 IK 需要所有 4 个列表（lowerLimits、upperLimits、jointRanges、restPoses）。 否则将使用常规 IK。 只为具有它们的关节提供限制（跳过固定关节），因此长度是自由度的数量。 请注意，lowerLimits、upperLimits、jointRanges 很容易导致 IK 解决方案发生冲突和不稳定。 首先尝试使用广泛的范围和限制，仅使用休息姿势。
可选用的	upperLimits	list of floats [0..nDof]	任选使用空空间 IK 需要所有 4 个列表（lowerLimits、upperLimits、jointRanges、restPoses）。 否则将使用常规 IK。lowerLimit 和 upperLimit 指定关节限制
可选用的	jointRanges	list of floats [0..nDof]	任选使用空空间 IK 需要所有 4 个列表（lowerLimits、upperLimits、jointRanges、restPoses）。 否则将使用常规 IK。
可选用的	restPoses	list of floats [0..nDof]	任选使用空空间 IK 需要所有 4 个列表（lowerLimits、upperLimits、jointRanges、restPoses）。 否则将使用常规 IK。支持更接近给定静止姿势的 IK 解决方案
可选用的	jointDamping	list of floats [0..nDof]	jointDamping 允许使用关节阻尼因子调整 IK 解决方案
可选用的	solver	int	p.IK_DLS 或 p.IK_SDLS，阻尼最小二乘或选择性阻尼最小二乘，如 Samuel Buss 的论文“用于逆运动学的选择性阻尼最小二乘”中所述。

可选用的	currentPosition	list of floats [0..nDof]	关节位置列表。 默认情况下，PyBullet 使用身体的关节位置。 如果提供，则 targetPosition 和 targetOrientation 位于本地空间中！
可选用的	maxNumIterations	int	优化 IK 解决方案，直到目标和实际末端执行器位置之间的距离低于此阈值，或达到 maxNumIterations。 默认为 20 次迭代。
可选用的	residualThreshold	double	优化 IK 解决方案，直到目标和实际末端执行器位置之间的距离低于此阈值，或达到 maxNumIterations。
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

calculateInverseKinematics 返回每个自由度的关节位置列表，因此该列表的长度是关节的自由度数（跳过了基础关节和固定关节）。 有关示例，请参见 Bullet/examples/pybullet/inverse_kinematics.py。

默认情况下，IK 将细化解，直到目标末端执行器和实际末端执行器之间的距离低于残差阈值 (1e-4) 或达到最大迭代次数。

6.3.3 calculateInverseKinematics2

类似于 calculateInverseKinematics，但它需要一个末端执行器索引列表及其目标位置（目前没有方向）。

必要的	bodyUniqueId	int	正文唯一 ID，由 loadURDF 返回
必要的	endEffectorLinkIndices	list of int	末端执行器链接索引
必要的	targetPositions	list of vec3	末端执行器的目标位置（它的链接坐标，而不是质心坐标！）。 默认情况下，这是在笛卡尔世界空间中，除非您提供 currentPosition 关节角度。7
...	有关其他参数，请参阅 calculateInverseKinematics		

7. Reinforcement Learning Gym Envs（强化学习 Gym Envs）

在“pip install pybullet”期间安装了一套 RL Gym Environments。 这包括 OpenAI Gym 环境的 PyBullet 版本，例如 ant、hopper、humanoid 和 walker。 还有一些环境适用于模拟以及真实机器人，例如 Ghost Robotics Minitaur 四足机器人、MIT 赛车和 KUKA 机器人手臂抓取环境。

pybullet 、 pybullet_envs 、 pybullet_data 的源代码和示例在这里：https://github.com/bulletphysics/bullet3/tree/master/examples/pybullet/gym.

您可以使用 RL 训练算法（例如 DQN、PPO、TRPO 和 DDPG）来训练环境。 有几个预训练的示例可用，您可以像这样享受它们：

pip install pybullet, tensorflow, gym

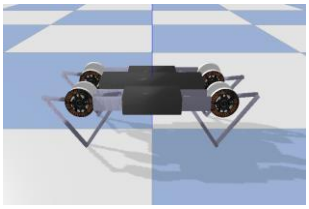
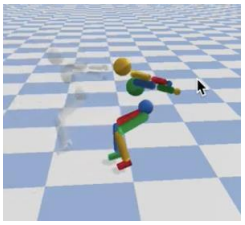
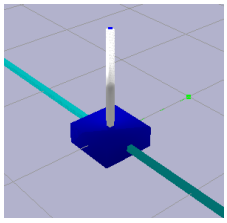
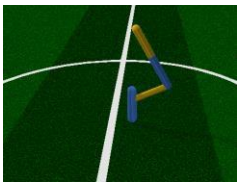
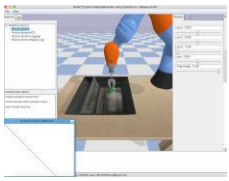
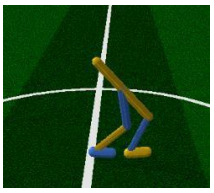
python -m pybullet_envs.examples.enjoy_TF_HumanoidBulletEnv_v0_2017may python -m pybullet_envs.examples.kukaGymEnvTest

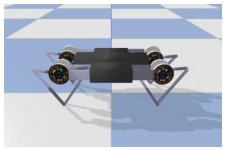
7.1 Environments and Data（环境和数据）

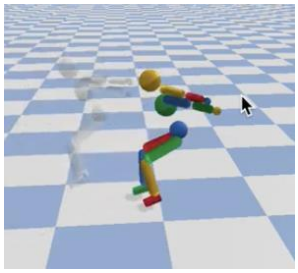
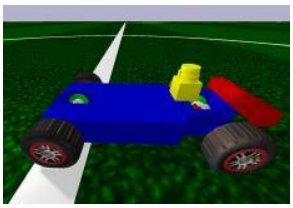
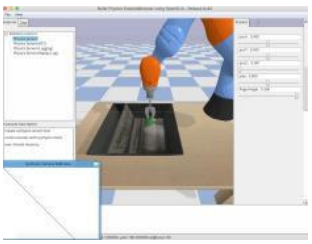
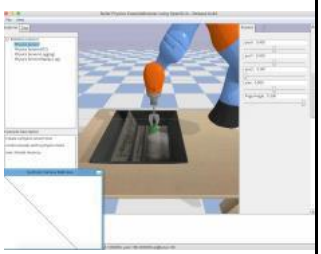
在“sudo pip install pybullet”之后，pybullet_envs 和 pybullet_data 包可用。 导入 pybullet_envs 包会自动将环境注册到 OpenAI Gym。

您可以使用以下 Python 行获取GYM中的 Bullet 环境列表：

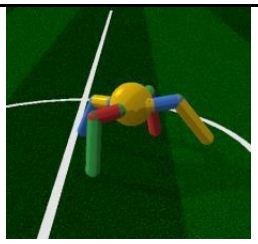
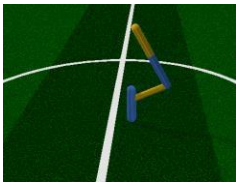
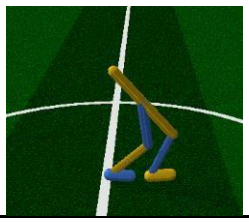
```
print(
```

MinitaurBulletEnv-v0 	HumanoidDeepMimic*BulletEnv-v1 	CartPoleContinuousBulletEnv-v0 
HumanoidBulletEnv-v0 	AntBulletEnv-v0 	HopperBulletEnv-v0 
KukaBulletEnv-v0 	HalfCheetahBulletEnv-v0 	Walker2DBulletEnv-v0 

Environment Name	Description
MinitaurBulletEnv-v0 	Ghost Robotics Minitaur 的模拟在平坦的地面上四足行走。 此环境用于 RSS 2018 “Sim-to-Real: Learning Agile Locomotion For Quadruped Robots”，请参阅 arxiv 上的论文。 根据行驶距离奖励。 使用 Gym 创建类： env = gym.make('MinitaurBulletEnv-v0') 或者直接使用类创建环境，带有参数： 导入 pybullet_envs.bullet.minitaur_gym_env 作为 e env = e.MinitaurBulletEnv(render=True)
HumanoidDeepMimic Backf lipBulletEnv-v1 and HumanoidDeepMimicWalk BulletEnv-v1	PyBullet 中 DeepMimic 论文的重新实现：模拟人形模拟参考运动。该实现允许选择参考运动。 后空翻和步行参考运动变成了 Gym 环境。 各种运动的预训练模型可作为 PyBullet 的一部分：需要 Tensorflow 1.x (1.14)： python3 -m pybullet_envs.deep_mimic.testrl --arg_file run_humanoid3d_backflip_args.txt

见视频VIDEO。以及使用 Human 3.6 数据集的示例 example 。 	python3 -m pybullet_envs.deep_mimic.testrl --arg_file run_humanoid3d_walk_args.txt 您还可以使用包含的 DeepMimic MPI 并行 PPO 实现进行训练： python3 mpi_run.py --arg_file train_humanoid3d_walk_args.txt --num_workers 16 （根据您的机器更改核心数 16）为了回放您训练的策略，您需要复制权重。
RacecarBulletEnv-v0 	MIT RC Racecar 的模拟。 根据与随机放置的球的距离奖励。 观察结果是相机帧中的球位置 (x,y)。 环境的动作空间可以是离散的（对于 DQN）或连续的（对于 PPO、TRPO 和 DDPG）。 import pybullet_envs.bullet.minitaur_gym_env as e env = e.RacecarGymEnv(isDiscrete=False ,renders=True) env.reset()
RacecarZedBulletEnv-v0 	与 RacecarBulletEnv-v0 相同，但观察值是相机像素。
KukaBulletEnv-v0 	模拟 KUKA Iiwa 机械臂，抓取托盘中的物体。 主要奖励发生在结束时，当被抓住的人可以抓住超过一定高度的物体时。 每一步都会发生一些非常小的奖励/成本：动作成本和抓手与物体之间的距离。 观察包括对象的 x,y 位置。 注意：这个环境目前训练有问题，我们会调查它。
KukaCamBulletEnv-v0 	与 KukaBulletEnv-v0 相同，但观察是相机像素。

我们将Roboschool environments 移植到 pybullet。 Roboschool 环境比 MuJoCo Gym 环境更难。

AntBulletEnv-v0	蚂蚁更重，鼓励它通常有两条或更多条腿
	地面。
HalfCheetahBulletEnv-v0 	
HumanoidBulletEnv-v0 	人形机器人受益于从奖励中减去的更现实的能量成本（= 扭矩 × 角速度）。
HopperBulletEnv-v0 	
Walker2DBulletEnv-v0 	
InvertedPendulumBulletEnv-v0	
InvertedDoublePendulumBulletEnv-v0	
InvertedPendulumSwingupBulletEnv-v0	

也可以从 pybullet_data 包中访问数据，例如 URDF/SDF 机器人资产、Wavefront .OBJ 文件。这是一个如何执行此操作的示例：

```
import pybullet
 import pybullet_data
datapath = pybullet_data.getDataPath()
pybullet.connect(pybullet.GUI)
pybullet.setAdditionalSearchPath(datapath)
pybullet.loadURDF("r2d2.urdf",[0,0,1])
```

也可以从 pybullet_data 包中访问数据，例如 URDF/SDF 机器人资产、Wavefront .OBJ 文件。这是一个如何执行此操作的示例：

7.2 Stable Baselines & ARS, ES,...(稳定的基线和 ARS、ES、...)

对于HalfCheetah（HalfCheetahBulletEnv-v0）、Ant（AntBulletEnv_v0）、（Hopper）HopperBulletEnv_v0、CartPoleContinuousBulletEnv-v0等连续控制Gym环境，可以使用 Stable Baselines。 这是一个例子：

pip3 install stable_baselines --
user pip3 install pybullet --user

python3 -m pybullet_envs.stable_baselines.train --algo sac --env HalfCheetahBulletEnv-v0

To enjoy the trained environment, copy/rename the weights file to sac_HalfCheetahBulletEnv- v0.zip (remove the _best part)

python3 -m pybullet_envs.stable_baselines.enjoy --algo sac --env HalfCheetahBulletEnv-v0 --n- episodes 5

Stable Baselines Zoo 提供预训练的 PyBullet 环境。

您还可以在 Google Colab 笔记本中使用 Stable Baselines 训练和享受 PyBullet 环境，请参阅此 Colab example of training a cartpole.

7.2.1 Train and Enjoy: DQN, PPO, ES(训练和奖赏：DQN、PPO、ES)

对于 KukaBulletEnv-v0 和 RacecarBulletEnv-v0 等离散 Gym 环境，您可以使用OpenAI Baselines DQN 使用离散动作空间训练模型。 提供了一些示例，如何训练和享受这些离散环境：

python -m pybullet_envs.baselines.train_pybullet_cartpole
python -m pybullet_envs.baselines.train_pybullet_racecar

当模型改进时，OpenAI Baselines 将在指定的时间间隔保存一个 .PKL 文件。 此 .PKL 文件用于奖励脚本：

python -m pybullet_envs.baselines.enjoy_pybullet_cartpole

python -m pybullet_envs.baselines.enjoy_pybullet_racecar

PyBullet 还附带了一些预训练模型，您可以开箱即用。 以下是可以奖励的预训练环境列表：

python -mpybullet_envs.examples.enjoy_TF_AntBulletEnv_v0_2017may

python -m pybullet_envs.examples.enjoy_TF_HalfCheetahBulletEnv_v0_2017may
python -m pybullet_envs.examples.enjoy_TF_AntBulletEnv_v0_2017may

python -m pybullet_envs.examples.enjoy_TF_HopperBulletEnv_v0_2017may
python -m pybullet_envs.examples.enjoy_TF_HumanoidBulletEnv_v0_2017may

python -m pybullet_envs.examples.enjoy_TF_InvertedDoublePendulumBulletEnv_v0_2017may
python -m pybullet_envs.examples.enjoy_TF_InvertedPendulumBulletEnv_v0_2017may

python -m pybullet_envs.examples.enjoy_TF_InvertedPendulumSwingupBulletEnv_v0_2017may
python -m pybullet_envs.examples.enjoy_TF_Walker2DBulletEnv_v0_2017may

7.2.2 Train using TensorFlow & PyTorch（使用 TensorFlow 和 PyTorch 进行训练）

您可以使用 TensorFlow Agents PPO 训练各种 pybullet 环境。 首先安装必备的Python包：pip install gym、tensorflow、agents、pybullet、ruamel.yaml

然后用于训练：

python -m pybullet_envs.agents.train_ppo --config=pybullet_pendulum --logdir=pendulum

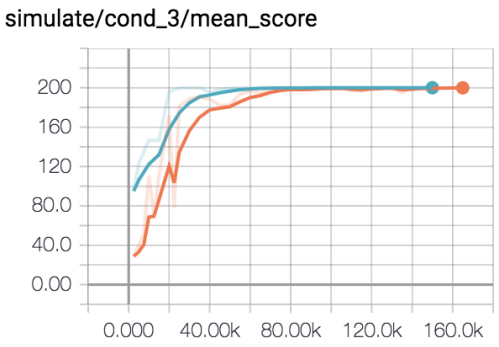
以下环境可用作代理配置：

pybullet_pendulum
pybullet_doublependulum
pybullet_pendulumswingup
pybullet_cheetah
pybullet_ant
pybullet_racecar
pybullet_minitaur

您可以使用 tensorboard 查看训练的进度：

tensorboard --logdir=pendulum --port=2222

打开网络浏览器并访问 localhost:2222 页面。 这是来自 Tensorboard 的用于钟摆训练的示例图：



训练后，您可以可视化已训练的模型、创建视频或使用物理服务器（python -m pybullet_envs.examples.runServer 或 ExampleBrowser 在物理服务器模式或虚拟现实中）进行可视化。 如果您启动本地 GUI 物理服务器，可视化器 (bullet_client.py) 将自动连接到它，并使用 OpenGL 硬件渲染来创建视频。 否则它将使用 CPU tinyrenderer。 要生成视频，请使用：

python -m pybullet_envs.agents.visualize_ppo --logdir=pendulum/xxxxx --outdir=pendulum_video 以类似的方式，您可以训练和可视化 Minitaur 机器人：

python -m pybullet_envs.agents.train_ppo --config=pybullet_minitaur --logdir=pybullet_minitaur 下面是 Minitaur 步态的示例视频：

https://www.youtube.com/watch?v=tfqCHDoFHRQ

7.2.3 Evolution Strategies (ES)（进化策略（ES））

David Ha (hardmaru) 在http://blog.otoro.net/2017/11/12/evolving-stable-strategies上有一篇关于如何使用 Evolution Strategies 训练 PyBullet 环境的博客文章

7.2.4 Train using PyTorch PPO（使用 PyTorch PPO 进行训练）

我们将添加一些描述如何开始使用 PyTorch 和 pybullet。 同时，请参阅此存储库：https://github.com/ikostrikov/pytorch-a2c-ppo-acktr

8. Virtual Reality（虚拟现实）

另请参阅 vrBullet 快速入门指南 vrBullet quickstart guide.

VR 物理服务器使用 OpenVR API 来支持 HTC Vive 和 Oculus Rift Touch 控制器。 OpenVR 目前在 Windows 上工作，Valve 也在Linux version上工作。

另请参阅

https://www.youtube.com/watch?v=VMJyZtHQL50 以获取 VR 示例的示例视频，它是 Bullet 的一部分，可以通过共享内存、UDP 或 TCP 连接使用 PyBullet 完全控制。

对于 Windows 上的 VR，建议使用 Microsoft Visual Studio (MSVC) 编译 Bullet Physics SDK。通过运行“build_visual_studio_vr_pybullet_double.bat”脚本生成 MSVC 项目文件。 您可以自定义这个小脚本以指向 Python 等的位置。确保切换到 MSVC 的“发布”配置并构建并运行 App_PhysicsServer_SharedMemory_VR*.exe。 默认情况下，此 VR 应用程序将呈现一个显示跟踪器/控制器（如果可用）的空世界。

8.1 getVREvents,setVRCameraState（获取VREvents，设置VRCameraState）

getVREvents 将返回自上次调用 getVREvents 以来已更改状态的选定 VR 设备的列表事件。不提供任何 deviceTypeFilter 时，默认只报告 VR_DEVICE_CONTROLLER 状态。 您可以选择任何设备组合，包括 VR_DEVICE_CONTROLLER、VR_DEVICE_HMD（头戴式设备）和 VR_DEVICE_GENERIC_TRACKER

（例如 HTC Vive 追踪器）。

请注意，VR_DEVICE_HMD 和 VR_DEVICE_GENERIC_TRACKER 仅报告位置和方向事件。

getVREvents 有以下参数：

可选用的	deviceTypeFilter	int	默认为 VR_DEVICE_CONTROLLER 您也可以选择 VR_DEVICE_HMD 或 VR_DEVICE_GENERIC_TRACKER 或它们的任意组合。
可选用的	allAnalogAxes	int	1 表示所有模拟轴，0 表示只有一个轴
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

输出参数为：

controllerId	int	controller index (0..MAX_VR_CONTROLLERS)
controllerPosition	vec3, list of 3 floats	controller position, in world space Cartesian coordinates
controllerOrientation	vec4, list of 4 floats	controller orientation quaternion [x,y,z,w] in world space
controllerAnalogueAxis	float	analogue axis value
numButtonEvents	int	number of button events since last call to getVREvents
numMoveEvents	int	number of move events since last call to getVREvents
buttons	int[64], list of button states (OpenVR has a maximum of 64 buttons)	flags for each button: VR_BUTTON_IS_DOWN (currently held down), VR_BUTTON_WAS_TRIGGERED (went down at least once since last cal to getVREvents, VR_BUTTON_WAS_RELEASED (was released at least once since last call to getVREvents). Note that only VR_BUTTON_IS_DOWN reports actual current state. For example if the button went down and up, you can tell from the RELEASE/TRIGGERED flags, even though IS_DOWN is still false. Note that in the log file, those buttons are packed with 10 buttons in 1 integer (3 bits per button).
deviceType	int	type of device: VR_DEVICE_CONTROLLER, VR_DEVICE_HMD or VR_DEVICE_GENERIC_TRACKER
allAnalogAxes (only if explicitly requested!)	list of 10 floats	currently, MAX_VR_ANALOGUE_AXIS is 5, for each axis x and y value.

有关 VR 绘图的示例，请参阅 Bullet/examples/pybullet/examples/vrEvents.py 和用于跟踪 HMD 和通用跟踪器的 Bullet/examples/pybullet/examples/vrTracker.py。

8.2 setVRCameraState（设置 VR 相机状态）

setVRCameraState 允许设置相机根变换偏移位置和方向。 这允许控制 VR 摄像机在虚拟世界中的位置。 也可以让 VR 相机跟踪物体，例如车辆。

setVRCameraState 有以下参数（没有返回值）：

可选用的	rootPosition	vec3, vector of 3 floats	相机根位置
可选用的	rootOrientation	vec4, vector of 4 floats	四元数组 [x,y,z,w] 格式的相机根方向。
可选用的	trackObject	vec3, vector of 3 floats	要跟踪的对象唯一 ID
可选用的	trackObjectFlag	int	flags.VR_CAMERA_TRACK_OBJECT_ORIENTATION（如果启用，位置和方向都会被跟踪）
可选用的	physicsClientId	int	如果您连接到多个服务器，则可以选择一个。

9. Debug GUI, Lines, Text, Parameters（调试 GUI、行、文本、参数）

PyBullet 具有一些功能，可以更轻松地调试、可视化和调整模拟。 此功能仅在有一些 3D 可视化窗口时有用，例如 GUI 模式或连接到单独的物理服务器（例如“物理服务器”模式下的示例浏览器或具有 OpenGL GUI 的独立物理服务器）。

9.1 addUserDebugLine, Text, Parameter（addUserDebugLine，文本，参数）

您可以添加由 3d 起点 (from) 和终点 (to)、颜色 [red,green,blue]、线宽和持续时间（以秒为单位）指定的 3d 线。 addUserDebugline 的参数是：

必要的	lineFromXYZ	vec3, list of 3 floats	笛卡尔世界坐标中直线的起点
必要的	lineToXYZ	vec3, list of 3 floats	笛卡尔世界坐标中直线的终点
可选用的	lineColorRGB	vec3, list of 3 floats	RGB 颜色 [Red, Green, Blue] 范围 [0..1] 中的每个分量
可选用的	lineWidth	float	线宽（受 OpenGL 实现限制）
可选用的	lifeTime	float	使用 0 表示永久线路，或以秒为单位的正时间（之后自动删除线路）
可选用的	parentObjectUniqueId	int	即将发布的 PyBullet 1.0.8 中的新功能：在父对象/链接的本地坐标中画线。
可选用的	parentLinkIndex	int	即将发布的 PyBullet 1.0.8 中的新功能：在父对象/链接的本地坐标中画线。
可选用的	replaceItemUniqueId	int	替换现有行（以提高性能并避免删除/添加闪烁）另见 f10_racecar.py 示例。
可选用的	physicsClientId	int	如果您连接到多台服务器，则可以选择一台

addUserDebugLine 将返回一个非负的唯一 id，让您可以使用 removeUserDebugItem 删除该行。（当使用“replaceItemUniqueId”时，它将返回 replaceItemUniqueId）。

9.1.1 addUserDebugText（添加用户调试文本）

您可以使用颜色和大小在特定位置添加一些 3d 文本。 输入参数是：

必要的	text	text	表示为字符串的文本（字符数组）

必要的	textPosition	vec3, list of 3 floats	笛卡尔世界坐标中文本的 3d 位置 [x,y,z]
可选用的	textColorRGB	vec3, list of 3 floats	RGB 颜色 [Red, Green, Blue] 范围 [0..1] 中的每个分量
可选用的	textSize	float	字体大小
可选用的	lifeTime	float	使用 0 表示永久文本，或以秒为单位的正时间（之后自动删除文本）
可选用的	textOrientation	vec4, list of 4 floats	默认情况下，调试文本将始终面向相机，自动旋转。 通过指定文本方向（四元数），方向将固定在世界空间或局部空间（当指定父级时）。 请注意，面向摄像机的文本使用不同的实现/着色器，具有不同的外观：面向摄像机的文本使用位图字体，具有指定方向的文本使用 TrueType 字体。
可选用的	parentObjectUniqueId	int	即将发布的 PyBullet 1.0.8 中的新功能：在父对象/链接的本地坐标中画线。
可选用的	parentLinkIndex	int	即将发布的 PyBullet 1.0.8 中的新功能：在父对象/链接的本地坐标中画线。
可选用的	replaceItemUniqueId	int	替换现有文本项（以避免删除/添加闪烁）
可选用的	physicsClientId	int	如果您连接到多台服务器，则可以选择一台

addUserDebugText 将返回一个非负的唯一 id，让您可以使用 removeUserDebugItem 删除该行。 另请参阅 pybullet/examples/debugDrawItems.py

9.1.2 addUserDebugParameter(添加用户调试参数)

addUserDebugParameter 允许您添加自定义滑块和按钮来调整参数。 它将返回一个唯一的 id。 这使您可以使用 readUserDebugParameter 读取参数的值。 addUserDebugParameter 的输入参数为：

必要的	paramName	string	参数名称
必要的	rangeMin	float	最小值。 如果最小值 > 最大值，将出现一个按钮而不是滑块
必要的	rangeMax	float	最大值
必要的	startValue	float	起始值
可选用的	physicsClientId	int	如果您连接到多台服务器，则可以选择一台

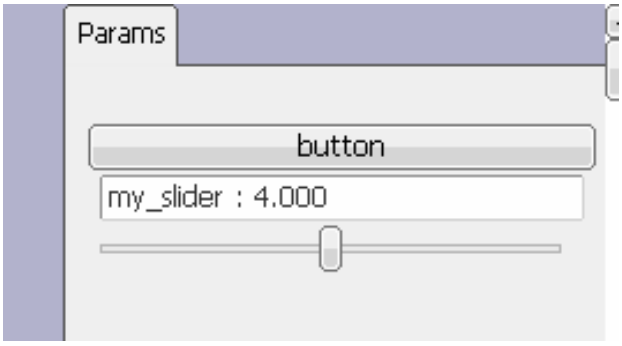
readUserDebugParameter 的输入参数为：

必要的	itemUniqueId	int	'addUserDebugParameter) 返回的唯一 ID
可选用的	physicsClientId	int	如果您连接到多台服务器，则可以选择一台

对于滑块，返回值是参数的最新读数。 对于按钮，按钮的 getUserDebugParameter 值在每次按下按钮时增加 1。

例子:

```
p.addUserDebugParameter("button",1,0,1)
p.addUserDebugParameter("my_slider",3,5,4)
```



9.1.3 removeAllUserParameters(删除所有用户参数)

这将删除所有滑块和按钮。

可选用的	physicsClientId	int	如果您连接到多台服务器，则可以选择一台

9.1.4 removeUserDebugItem/All(删除用户调试项/全部)

添加用户调试行的函数，如果成功，文本将返回一个非负的唯一 id。 您可以使用 removeUserDebugItem 方法使用此唯一 id 删除调试项。输入参数为：

必要的	itemUniqueId	int	要删除的调试项的唯一 ID（行、文本等）
可选用的	physicsClientId	int	如果您连接到多台服务器，则可以选择一台

9.1.5 removeAllUserDebugItems(删除用户调试项)

此 API 将删除所有调试项（文本、行等）。

9.1.6 setDebugObjectColor(设置调试对象颜色)

内置的 OpenGL 可视化器具有线框调试渲染功能：按“w”进行切换。 线框有一些默认颜色。 您可以使用 setDebugObjectColor 覆盖特定对象和链接的颜色。 输入参数为：

必要的	objectUniqueId	int	对象的唯一标识
必要的	linkIndex	int	链接索引
可选用的	objectDebugColorRGB	vec3, list of 3 floats	[Red,Green,Blue] 中的调试颜色。 如果未提供，自定义颜色将被删除。
可选用的	physicsClientId	int	如果您连接到多台服务器，则可以选择一台

9.2 addUserData(添加用户数据)

简而言之，添加、删除和查询用户数据，目前是文本字符串，附加到正文的任何链接。 请参阅 userData.py 示例以了解如何使用它。 它将返回一个 userDataId。 请注意，您还可以在 urdf 文件中添加用户数据。

9.2.1 getUserData(获取用户数据)

getUserData 将根据 addUserData 返回的 userDataId 接收用户数据。 有关示例用法，请参见 userData.py。

9.2.2 syncUserData(同步用户数据)

如果多个客户端更改用户数据（addUserData 等），syncUserData 将同步用户数据（getUserData）。

9.2.3 removeUserData(删除用户数据)

给定 userDataId，removeUserData 将删除之前添加的用户数据。

9.2.4 getUserDataId and getNumUserData(获取用户数据 ID 并获取用户数据数量)

getNumUserData 将返回给定 bodyUniqueId 的用户数据条目数。

9.2.5 getUserDataInfo(获取用户数据信息)

getUserDataInfo 检索用户数据的键和标识符为 (userDataId, key, bodyUniqueId, linkIndex, visualShapeIndex)

9.3 configureDebugVisualizer(配置调试可视化器)

您可以配置内置 OpenGL 可视化器的一些设置，例如启用或禁用线框、阴影和 GUI 渲染。 这很有用，因为某些笔记本电脑或桌面 GUI 在我们的 OpenGL 3 可视化器中存在性能问题。

必要的	flag	int	特征启用或禁用，如 COV_ENABLE_WIREFRAME，COV_ENABLE_SHADOWS，COV_ENABLE_GUI，COV_ENABLE_VR_PICKING，COV_ENABLE_VR_TELEPORTING，COV_ENABLE_RENDERING，COV_ENABLE_TINY_RENDERER，COV_ENABLE_VR_RENDER_CONTROLLERS，COV_ENABLE_KEYBOARD_SHORTCUTS，COV_ENABLE_MOUSE_PICKING，COV_ENABLE_Y_AXIS_UP（Z是默认世界向上轴），COV_ENABLE_RGB_BUFFER_PREVIEW，COV_ENABLE_DEPTH_BUFFER_PREVIEW，COV_ENABLE_SEGMENTATION_MARK_PREVIEW
必要的	enable	int	0 or 1
可选用的	lightPosition	vec3	可视化仪的灯光位置
可选用的	shadowMapResolution	int	阴影贴图纹理的大小，对于许多 GPU，通常是 2 的幂。 默认值为 4096。现代 GPU 可以处理 16384 或 32768 或更高。
可选用的	shadowMapWorldSize	int	世界空间中阴影贴图的大小（以米为单位，默认为 10）
可选用的	physicsClientId	int	如果您连接到多台服务器，则可以选择一台

例如:

pybullet.configureDebugVisualizer(pybullet.COV_ENABLE_WIREFRAME,1)

9.4 get/resetDebugVisualizerCamera（获取/重置调试展示台相机）

警告：getDebugVisualizerCamera 的返回参数与 resetDebugVisualizerCamera 的顺序不同。将在未来的 API 修订版（主要新版本）中修复。

9.4.1 resetDebugVisualizerCamera（重置调试展示台相机）

您可以重置 3D OpenGL 调试可视化器相机距离（眼睛和相机目标位置之间）、相机偏航和俯仰以及相机目标位置。

必要的	cameraDistance	float	眼睛到相机目标位置的距离
必要的	cameraYaw	float	相机偏航角（以度为单位）左/右
必要的	cameraPitch	float	相机俯仰角（以度为单位）向上/向下
必要的	cameraTargetPosition	vec3, list of 3 floats	cameraTargetPosition 是相机焦点
可选用的	physicsClientId	int	如果您连接到多台服务器，则可以选择一台

示 例 ： pybullet.resetDebugVisualizerCamera(cameraDistance=3, cameraYaw=30, cameraPitch=52, cameraTargetPosition=[0,0,0])

9.4.2 getDebugVisualizerCamera（获取调试展示台相机）

您可以使用此命令获取相机的宽度和高度（以像素为单位）、其视图和投影矩阵。 输入参数是可选用的physicsClientId。 输出信息为：

width	int	相机图像的宽度（以像素为单位）
height	int	相机图像的高度（以像素为单位）
viewMatrix	float16, list of 16 floats	摄像机的视图矩阵
projectionMatrix	float16, list of 16 floats	相机的投影矩阵
cameraUp	float3, list of 3 floats	相机的上轴，在笛卡尔世界空间坐标中
cameraForward	float3, list of 3 floats	相机的前轴，在笛卡尔世界空间坐标中
horizontal	float3, list of 3 floats	待定。 这是一个水平向量，可用于生成光线（例如，用于鼠标拾取或创建简单的光线追踪器）
vertical	float3, list of 3 floats	待定。这是一个垂直向量，可用于生成光线（例如，用于鼠标拾取或创建简单的光线追踪器）。
yaw	float	相机的偏航角，在笛卡尔局部空间坐标中
pitch	float	相机的俯仰角，在笛卡尔局部空间坐标中
dist	float	相机和相机目标之间的距离
target	float3, list of 3 floats	相机的目标，在笛卡尔世界空间坐标中

9.5 getKeyboardEvents, getMouseEvents（获取键盘事件、获取鼠标事件）

您可以接收自上次调用“getKeyboardEvents”以来发生的所有键盘事件。 每个事件都有一个键码和一个状态。 状态是 KEY_IS_DOWN 、 KEY_WAS_TRIGGERED 和 KEY_WAS_RELEASED 的位标志组合。 如果键从“向上”状态变为“向下”状态，您将收到 KEY_IS_DOWN 状态以及 KEY_WAS_TRIGGERED 状态。 如果一个键被按下和释放，状态将为 KEY_IS_DOWN 和 KEY_WAS_RELEASED。

定义了一些特殊键：B3G_F1 ... B3G_F12、B3G_LEFT_ARROW、B3G_RIGHT_ARROW、B3G_UP_ARROW 、 B3G_DOWN_ARROW 、 B3G_PAGE_UP 、 B3G_PAGE_DOWN 、 B3G_PAGE_END 、 B3G_HOME 、 B3G_DELETE 、 B3G_INSERT 、 B3G_ALT 、 B3G_SHINURRET、B3G。

getKeyboardEvents的输入量是一个可选的物理客户端 ID：

可选用的	physicsClientId	int	如果您连接到多台服务器，则可以选择一台

输出是键码“键”和键盘状态“值”的字典。 例如

```
qKey = ord('q')
keys = p.getKeyboardEvents()
if qKey in keys and keys[qKey]&p.KEY_WAS_TRIGGERED: break;
```

9.5.4 getMouseEvents（获取鼠标事件）

与 getKeyboardEvents 类似，您可以获取自上次调用 getMouseEvents 以来发生的鼠标事件。 所有鼠标移动事件都合并为具有最新位置的单个鼠标移动事件。 此外，合并给定按钮的所有鼠标按钮事件。 如果按钮上下移动，状态将为“KEY_WAS_TRIGGERED”。 我们将 KEY_WAS_TRIGGERED /KEY_IS_DOWN /KEY_WAS_RELEASED 重用于鼠标按钮状态。

getMouseEvents 的输入参数是：

可选用的	physicsClientId	int	如果您连接到多台服务器，则可以选择一台

输出是以下格式的鼠标事件列表：

eventType	int	MOUSE_MOVE_EVENT=1, MOUSE_BUTTON_EVENT=2
mousePosX	float	鼠标指针的 x 坐标
mousePosY	float	鼠标指针的 y 坐标
buttonIndex	int	鼠标左/中/右键的按钮索引
buttonState	int	标志 KEY_WAS_TRIGGERED /KEY_IS_DOWN /KEY_WAS_RELEASED

有关鼠标事件的示例，请参见createVisualShape.py ，以选择/着色对象。

10. Plugins（插件）

PyBullet 允许您使用 C 或 C++ 编写插件以添加自定义功能。 PyBullet 的一些核心功能被编写为插件，例如 PD 控制、渲染、gRPC 服务器、碰撞过滤和虚拟现实同步。 作为 PyBullet 核心部分的大多数插件默认情况下都是静态链接的，因此您无需手动加载和卸载它们。

在 Linux 上，eglPlugin 是默认情况下与 PyBullet 一起提供的插件示例。 它可以启用在没有 X11 上下文的情况下使用硬件 OpenGL 3.x 渲染，例如在 Google Cloud Platform 上进行云渲染。 请参阅 eglRenderTest.py 示例如何使用它。

PyBullet 还带有一个 fileIOPlugin，它可以直接从 zip 文件加载文件并允许文件缓存。 请参阅 fileIOPlugin.py 示例如何使用它。

10.1 loadPlugin,executePluginCommand（加载插件，执行插件命令）

您可以使用 loadPlugin 命令加载 PyBullet 插件：

必要的	pluginPath	string	路径，磁盘上的位置在哪里可以找到插件
必要的	postFix	string	附加到每个 API 的插件的后缀名称
可选用的	physicsClientId	int	如果您连接到多台服务器，则可以选择一台

loadPlugin 将返回一个 pluginUniqueId 整数。 如果此 pluginId 为负，则不加载插件。 加载插件后，您可以使用以下命令向插件发送命令

10.1.1 executePluginCommand（执行插件命令）：

必要的	pluginUniqueId	int	插件的唯一 id，由 loadPlugin 返回
可选用的	textArgument	string	可选文本参数，由插件解释
可选用的	intArgs	list of int	可选的整数列表，由插件解释
可选用的	floatArgs	list of float	可选的浮动列表，由插件解释
可选用的	physicsClientId	int	如果您连接到多台服务器，则可以选择一台

10.2 unloadPlugin（卸载插件）

您可以使用 pluginId 卸载插件。

插件 API 与 PyBullet 共享相同的底层 C API，并具有与 PyBullet 相同的功能。

您可以浏览plugin implementation 实现以了解可能的情况。

11. Build and install PyBullet（构建和安装 PyBullet）

在 Windows、Mac OSX 和 Linux 上安装 PyBullet 有几种不同的方法。 我们使用 Python 2.7 和 Python 3.5.2，但预计大多数 Python 2.x 和 Python 3.x 版本应该可以工作。 让 PyBullet 工作的最简单方法是使用 pip 或 python setup.py：

11.1 Using Python pip

确保 Python 和 pip 已安装，然后运行：

pip install pybullet

您可能需要使用 sudo pip install pybullet 或 pip install pybullet --user。

请注意，如果您使用 pip 安装 PyBullet，安装 C++ Bullet Physics SDK 仍然有益：它包括数据文件、物理服务器和对 PyBullet 有用的工具。

您还可以在 Bullet Physics SDK 的根目录中运行“python setup.py build”和“python setup.py install”（从 http://github.com/bulletphysics/bullet3 获取 SDK）

另请参阅 https://pypi.python.org/pypi/pybullet

或者，您可以使用 premake (Windows) 或 cmake 从源代码安装 PyBullet：

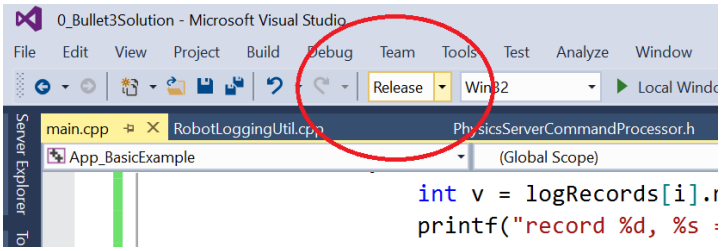
11.2 Using premake for Windows（在 Windows 上使用 premake）

确保在 c:\python-3.5.2 （或其他版本文件夹名称）中安装了一些 Python 版本

首先从github获取源代码，使用git clone https://github.com/bulletphysics/bullet3

单击 build_visual_studio_vr_pybullet_double.bat 并在 Visual Studio 中打开 0_Bullet3Solution.sln 项目，如果需要转换项目。

切换到发布模式，并编译“pybullet”项目。



然后有几个选项可以在 Python 解释器中导入 pybullet：

1) 将 pybullet_vs2010.dll 重命名为 pybullet.pyd 并使用 bullet/bin 作为当前工作目录启动 Python.exe 解释器。 任选用于调试：将 bullet/bin/pybullet_vs2010_debug.dll 重命名为 pybullet_d.pyd 并启动 python_d.exe)Rename bullet/bin/pybullet_vs2010..dll to pybullet.pyd and use command prompt: set PYTHONPATH=c:\develop\bullet3\bin (replace with actual folder where Bullet is located) or create this PYTHONPATH environment variable using Windows GUI

2) 创建管理员提示符（cmd.exe）并创建符号链接如下 cd c:\python-3.5.2\dlls

mklink pybullet.pyd c:\develop\bullet3\bin\pybullet_vs2010.dll 然后运行 python.exe 并导入 pybullet 应该可以工作。

11.3 Using cmake on Linux and Mac OSX（在 Linux 和 Mac OSX 上使用 cmake）

请注意，推荐的方法是使用 sudo pip install pybullet（或 pip3）。 使用 cmake 或 premake 或其他构建系统仅适用于知道自己在做什么的开发人员，并且通常不受支持。

首先从github获取源代码，使用git clone https://github.com/bulletphysics/bullet3

1. 下载并安装 cmake

2. 在 Bullet 根目录运行 shell 脚本： build_cmake_pybullet_double.sh

3. 确保 Python 找到我们的 pybullet.so 模块：export PYTHONPATH = /your_path_to_bullet/build_cmake/examples/pybullet

按照这样。 通过运行 python 解释器测试 pybullet 并输入“import pybullet”以查看模块是否加载。 如果是这样，您可以使用 Bullet/examples/pybullet 中的 pybullet 脚本。

11.4 Possible Mac OSX Issues（可能的 Mac OSX 问题）

- 如果您在 Mac OSX 上导入 pybullet 时遇到任何问题，请确保您运行正确的 Python 解释器，匹配在 -DPYTHON_INCLUDE_DIR 和 -DPYTHON_LIBRARY 中设置的包含/库（使用 cmake）。 您可能安装了多个 Python 解释器，例如在使用自制软件时。 有关示例，请参见此评论。

- 尝试使用 CFLAGS='-stdlib=libc++' pip install pybullet，看到这个 issue.

11.5 Possible Linux Issues（可能的 Linux 问题）

- 确保已安装 OpenGL

- 当使用 Anaconda 作为 Python 发行版时，conda install libgcc 以便找到“GLIBCXX”（参见 http://askubuntu.com/questions/575505/glibcxx-3-4-20-not-found-how-to-fix- this-error)

- 使用Anaconda 作为Python 发行版时，cmake 可能找不到python 库。 您可以通过转到 ../build_cmake/CMakeCache.txt 文件并更改以下行来手动添加它们：'PYTHON_LIBRARY:FILEPATH=/usr/lib/python2.7/config-x86_64-linux-gnu/libpython2.7.so '

11.6 GPU or virtual machine lacking OpenGL 3（缺少 OpenGL 3 的 GPU 或虚拟机）

- 默认情况下，PyBullet 使用 OpenGL 3。一些远程桌面环境和 GPU 不支持 OpenGL 3，导致伪影（灰屏）甚至崩溃。 您可以使用 --opengl2 标志回退到 OpenGL 2。这不完全受支持，但它为您提供了一些查看场景的方法。

    - pybullet.connect(pybullet.GUI,options="--opengl2")

- 或者，您可以在远程机器上运行物理服务器，使用 UDP 或 TCP 桥接器，并通过 UDP 隧道从本地笔记本电脑连接到远程服务器。（待办事项：详细描述步骤）

11.7 Support, Tips, Citation（支持、提示、引用）

问题：我们在哪里寻求支持和报告问题?

Answer: There is a discussion forum at http://pybullet.org/Bullet and an issue tracker at https://github.com/bulletphysics/bullet3

Question: 我们如何在我们的学术论文中添加对 PyBullet 的引用？

回答：

```
@MISC{coumans2020,
    author = {Erwin Coumans and Yunfei Bai},
    title = {PyBullet, a Python module for physics simulation for games, robotics and machine learning},
    howpublished =
```

{\url{http://pybullet.org}}, year = {2016--2020}}

Question: PyBullet 可以在 Google Colab 中使用吗？

Answer: 是的，我们提供了可在 Colab 中使用的预编译 manylinux 轮子.

GPY 渲染也可以使用 EGL。 在此处查看示例 Colab。

问：Bullet 2.x 和 Bullet 3 OpenCL 实现会发生什么变化？

Answer:PyBullet 正在包装 Bullet C-API。 我们将把 Bullet 3 OpenCL GPU API（以及未来的 Bullet 4.x API）放在这个 C-API 后面。 因此，如果您使用 PyBullet 或 C-API，那么您是面向未来的。 不要与 Bullet 2.x C++ API 混淆。

Question:我应该使用扭矩/力控制还是速度/位置控制模式？

一般来说，最好从位置或速度控制开始。

要使力/扭矩控制可靠地工作需要付出更多的努力.

Question: 物体的速度似乎比预期的要小。 PyBullet 是否应用了一些默认阻尼？ 速度也不超过100个单位.

Answer: 是的，PyBullet 应用了一些角度和线性阻尼来增加稳定性。 您可以使用“changeDynamics”命令修改/禁用此阻尼，使用 linearDamping=0 和 angularDamping=0 作为参数.为了稳定性，最大线/角速度被限制在 100 个单位.

Question: 如何仅为机器人的某些部分（例如手臂）关闭重力？

回答：目前这还没有暴露出来，所以你需要为所有物体转动重力加速度，并为需要它的物体手动应用重力。 或者您可以主动计算重力补偿力，就像在真实机器人上发生的那样。 由于 Bullet 有一个完整的约束系统，因此计算这些反重力是微不足道的：您可以运行第二个模拟（PyBullet 允许您连接到多个物理服务器）并将机器人置于重力下，设置关节位置控制以保持 根据需要定位，并聚集那些“反重力”力量。 然后将它们应用到主模拟中。

Question: 如何放大/缩小对象?

Answer: 您可以将 globalScaleFactor 值用作 loadURDF 和 loadSDF 的可选参数。 否则，视觉形状和碰撞形状的缩放是大多数文件格式的一部分，例如 URDF 和 SDF。 目前您无法重新缩放对象。

Question:如何在模型中获取纹理？

Answer: 您可以使用 Wavefront .obj 文件格式。 这将支持材料文件 (.mtl)。Bullet/data 文件夹中有各种使用纹理的示例。 您可以使用“changeTexture”API 更改现有纹理对象的纹理。

Question:哪些纹理文件格式对 PyBullet 有效？

Answer: Bullet 使用 stb_image 加载纹理文件，加载 PNG、JPG、TGA、GIF 等。

有关详细信息，请参见 stb_image.h

Question: 如何提高碰撞检测的性能和稳定性？

答：优化的方法有很多，例如：形状类型

1) 选择一种或多种原始碰撞形状类型，如盒子、球体、胶囊、圆柱体来逼近一个物体，而不是使用凸或凹三角形网格。.

2) 如果您确实需要使用三角形网格，请使用分层近似凸分解（v-HACD）创建凸分解。 test_hacd utility将 OBJ 文件中的凸三角形网格转换为具有多个凸包对象的新 OBJ 文件。 参见例如 Bullet/data/teddy_vhacd.urdf 指向 Bullet/data/teddy2_VHACD_CHs.obj，或 duck_vhacd.urdf 指向 Duke_vhacd.obj

3) 减少三角形网格中的顶点数。 例如 Blender 3D 有一个很棒的网格抽取修改器，它可以交互地让你看到网格简化的结果.

4) 使用小的正值作为滚动摩擦力（例如 0.01）和旋转摩擦力用于圆形物体，例如球体和胶囊，以及使用 <link><contact> 节点内的 <rolling_friction> 和 <spinning_friction> 节点。 参见例如 Bullet/data/sphere2.urdf

5) 使用 <stiffness value="30000"/> 对车轮使用少量合规性 URDF <link><contact> xml 节点内的 <damping value="1000"/> 。 参见例如 Bullet/data/husky/husky.urdf 车辆.

6) 使用Bullet的双精度构建，这对接触稳定性和碰撞精度都有好处。选择一些好的约束求解器设置和时间步长.

7) 将物理模拟与图形解耦。 PyBullet 已经为 GUI 和各种物理服务器做到了这一点：OpenGL 图形可视化在它自己的线程中运行，独立于物理模拟.

Question: 摩擦处理有哪些选择？

Answer: 默认情况下，Bullet 和 PyBullet 对库仑摩擦模型使用精确的隐式圆锥摩擦。 此外，您可以通过在 <link><contact> 节点中添加 <rolling_friction> 和 <spinning_friction> 节点来启用滚动和旋转摩擦，例如，请参见 Bullet/data/sphere2.urdf 。 您可以启用金字塔近似，而不是圆锥摩擦。

Question: Bullet 内部有什么样的常数或阈值，使高速成为不可能？

Answer: 默认情况下，Bullet 依赖离散碰撞检测与穿透恢复相结合。 纯粹依靠离散碰撞检测意味着一个物体在一个时间步长内的行进速度不应超过其自身的半径。 PyBullet 使用 1./240 作为默认时间步长。 大于 1./60 的时间步长可能会由于各种原因（深度渗透、数值积分器）引入不稳定性。 Bullet has 是一种用于连续碰

撞检测的选项，用于捕捉在一个时间步内移动速度超过其自身半径的对象的碰撞。不幸的是，这种连续的碰撞检测可能会引入其自身的问题（性能和非物理响应、缺乏恢复），因此默认情况下不启用此实验功能。 查看 experimentalCcdSphereRadius.py 示例如何启用它

Question: 一些 API 没有记录。 通常这意味着（1）我们尚未更新快速入门指南或（2）该功能太实验性而无法记录。 如果您真的想了解特定的未记录 API，可以在跟踪器中提出问题。