

PyBullet快速入门指南

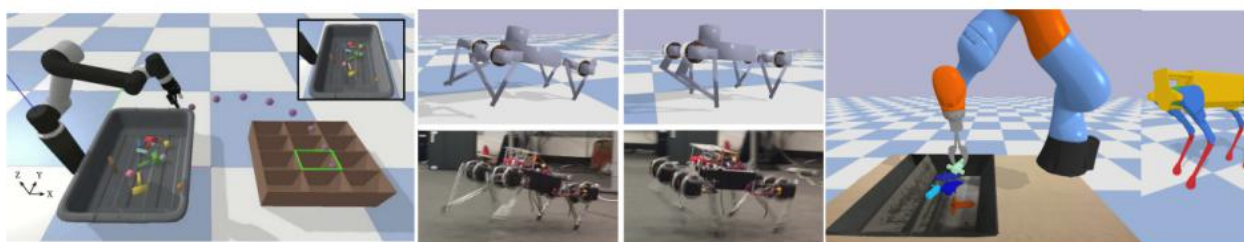
[欧文 云菲库曼斯](#) 白, 2016-2022

[访问桌面 博士, 论坛, github讨论和明星 子弹](#)

| | | |
|---|-----------------------------------|-----------|
| 《简介: 22 | 合成相机渲染 | 47 |
| 你好, 小子弹世界3 | computeView/ProjectionMatrix | 47 |
| 连接、断开连接, 子弹头_客户端3 | getCameraImage | 48 |
| setGravity 7 | getVisualShapeData | 51 |
| loadURDF, loadSDF, loadMJCF 7 | changeVisualShape, | 52 |
| saveState, saveBullet, restoreState 11 | loadTexture | 52 |
| createCollisionShape/VisualShape 12 | 碰撞检测查询 | 52 |
| createMultiBody 15 | getOverlappingObjects, getAABB | 53 |
| stepSimulation 16 | getContactPoints, | 55 |
| 获取基本位置和方向18 | getClosestPoints rayTest, | 56 |
| 重置底座位置和方向18 | rayTestBatch | 57 |
| 变换: 位置和方向19 | getCollisionShapeData | 60 |
| 控制机器人21 | 启用/禁用碰撞 | 60 |
| 底座、接头、连杆21 | 逆动力学, 运动学 | 60 |
| getNumJoints, getJointInfo 22 | 计算逆动力学 (2) 计算雅可比矩 | 62 |
| setJointMotorControl2/Array 23 | 阵, 大质量矩阵计算逆运动学 | 65 |
| getJointState (s), resetJointState 28 | (2) | 65 |
| 启用联合力扭矩传感器29 | 强化学习健身房 | 69 |
| getLinkState (s) 30 | 环境和数据稳定的基线和ARS, | 72 |
| getBaseVelocity, resetBaseVelocity 32 | ES,。 | 72 |
| applyExternalForce/Torque 32 | 虚拟现实 | 74 |
| getNumBodies, getBodyInfo, getBodyUniqueId, | getVREvents, setVRCameraState | 74 |
| removeBody 33 | | 77 |
| createConstraint, removeConstraint, | 调试GUI、行、文本、参数 | 78 |
| changeConstraint 33 | 添加用户调试程序, 行, 文本, 参数 | 78 |
| getNumConstraints, getConstraintUniqueId 35 | 添加用户数据 | 80 |
| getConstraintInfo/State 35 | configureDebugVisualizer | |
| getDynamicsInfo/changeDynamics 36 | get/resetDebugVisualizerCamera | 81 |
| setTimeStep 39 | getKeyboardEvents, getMouseEvents | 81 |
| setPhysicsEngineParameter 40 | 插件 | 82 |
| resetSimulation 42 | loadPlugin, executePluginCommand | 85 |
| startStateLogging/stopStateLogging 42 | 构建和安装PyBullet | |
| 变形和布 (FEM, PBD) 44 | 支持、提示、引文 | |
| loadSoftBody/loadURDF 45 | | |
| createSoftBodyAnchor 46 | | |

介绍

PyBullet是一个快速和易于使用的Python模块，用于机器人模拟和机器学习，专注于模拟到真实的传输。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，逆动力学计算，正向和逆运动学，碰撞检测和射线交叉查询。[子弹 物理学](#) SDK包括小子弹机器人的例子，如模拟的微型龙四足动物，使用张力流推理运行的类人动物和KUKA手臂抓取物体。简化坐标多体、刚性体和可变形体由统一的LCP约束求解器处理，类似于本文中的铰接岛算法，它使用铰接体算法进行线性时间正向动力学，并创建求解器a矩阵。



除了物理模拟之外，还有一些对渲染的绑定，包括一个CPU渲染器（修补程序渲染器）和OpenGL 3。x渲染和可视化和支持虚拟现实头盔，如HTC Vive和Oculus Rift。PyBullet还具有执行碰撞检测查询（最近点、重叠对、射线交叉测试等）和添加调试呈现（调试行和文本）的功能。PyBullet对共享内存、UDP和TCP网络提供了跨平台内置的客户端-服务器支持。所以你可以在Linux上运行子弹连接到Windows VR服务器。

小子弹包裹新的子弹_C-API，它被设计为独立于底层的物理引擎和渲染引擎，因此我们可以轻松地迁移到Bullet的新版本，或者使用不同的物理引擎或渲染引擎。默认情况下，小球使用子弹2。在CPU上的x API。我们将揭露子弹3。x运行在GPU上使用OpenCL以及。[还有一个类似于PyBullet的C++API，请参见b3机器人模拟器客户端API。](#)

小子弹可以很容易地使用张量流和OpenAI健身房。谷歌的研究人员[大脑\[1、2、3，4\]](#)，X[1,2]，斯坦福大学人工智能实验室[\[1、2、3\]、OpenAI、INRIA \[1\]和许多其他的](#)实验室使用PyBullet。如果你在你的研究中使用小子弹，请添加引文_。

PyBullet的安装很简单，比如(sudo) pip安装PyBullet (Python 2.x)，pip3安装PyBullet。这将暴露小子弹模块以及pybullet_envs Gym环境。

你好，PyBullet世界

下面是我们逐步讨论的一个PyBullet介绍脚本：

```

导入piture作为p
导入时间
导入pybullet_data
physicsClient = p.连接 (p. GUI)#or p. 直接用于非图形版本
p. 设置附加的搜索路径(pybullet_data.getDataPath ()) #optionally
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)
#设置质心帧(loadURDF设置基本链接帧)
startPos/Ornp. 重置底座位置和方向(箱号, startPos,
startOrientation)
范围 (10000) :
    p.stepSimulation ()
    睡眠时间 (1./240.)
cubePos, cubeOrn = p.getBasePositionAndOrientation (boxId)
打印 (cubePos、cubeOrn)
p. 切断

```

连接，断开连接，子弹头_客户端

在导入PyBullet模块后，要做的第一件事是“连接”到物理模拟。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：直接服务器和GUI服务器。GUI和直接连接都将在与PyBullet相同的过程中执行物理模拟和渲染。

请注意，在直接模式下，您无法访问OpenGL和VR硬件特性，如“虚拟现实”和“调试GUI、行、文本、参数”章节中所述。直接模式确实允许使用内置的软件渲染器通过“获取相机图像”API渲染图像。这对于在没有GPU的服务器上在云中运行模拟很有用。

您可以提供自己的数据文件，也可以使用附带的PyBullet_data软件包PyBullet。为此，导入pybullet_data并使用
`pybullet.setAdditionalSearchPath (pybullet_data.getDataPath())`。

getConnectionInfo

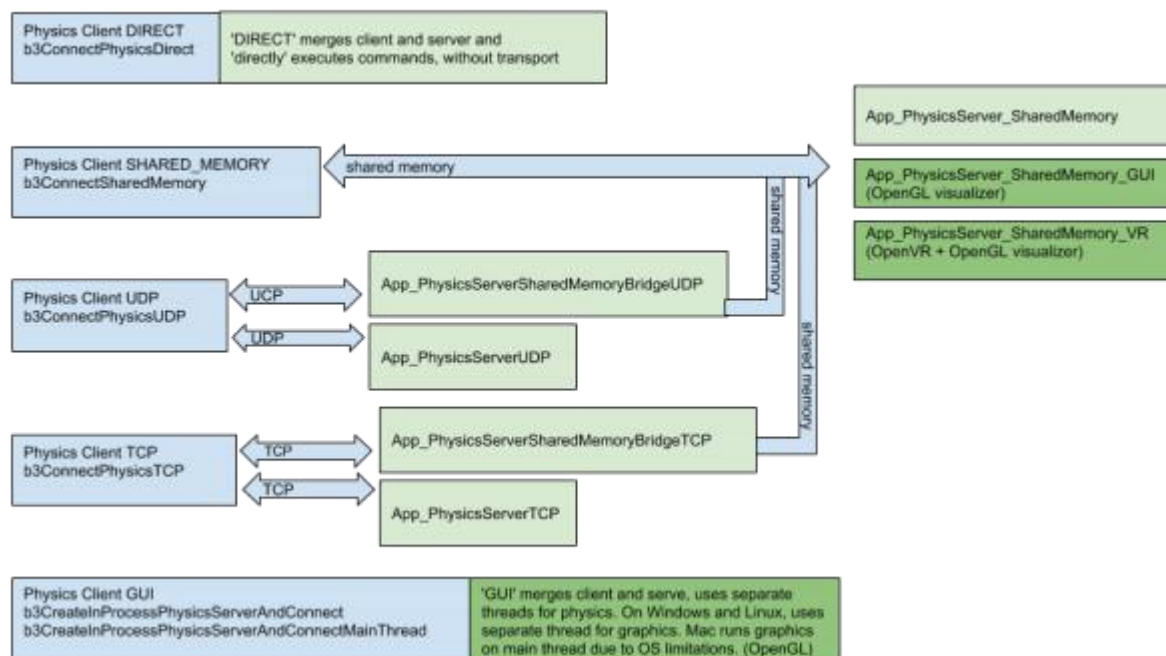
给定一个物理的，clientId将返回列表[已连接，连接方法]

已连接

如果连接将返回true，否则返回false，给定一个物理clientId。

设置超时

如果服务器在特定的超时时间值内没有处理一个命令，则客户端将断开连接。使用设置超时时间可以秒为单位指定此值。



各种物理客户端（蓝色）和物理服务器（绿色）选项。深绿色服务器提供了OpenGL调试可视化。

使用直接，GUI连接

直接连接将命令直接发送到物理引擎，不使用任何传输层，也没有图形可视化窗口，并在执行命令后直接返回状态。

GUI连接将创建一个新的图形用户界面（GUI），在与PyBullet相同的进程空间内。在Linux和Windows上，这个GUI运行在一个单独的线程中，而在OSX上，由于操作系统的限制，它运行在同一个线程中。在Mac OSX上，你可能会在OpenGL窗口中看到一个旋转轮，直到你运行一个“逐步模拟”或其他PyBullet命令。

命令和状态消息在PyBullet客户端和GUI物理模拟服务器之间发送。

也可以在同一台机器上或使用SHARED_MEMORY、UDP或TCP网络在远程机器上以不同的进程连接到物理服务器。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，该方法不解析关键字参数。

连接的输入参数包括：

| | | | |
|------|------------------|---|--|
| 必须的 | 连接方式 | 整数：直接，GUI，SHARED_内存，UDP，TCP GUI_SERVER，SHARED_MEMORY_SERVER，SHARED_MEMORY_图形用户界面 | 直接模式创建了一个新的物理引擎，并直接与之通信。GUI将创建一个具有图形化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，并通过共享内存与之通信。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类似于直接连接，但也作为一个允许外部SHARED_MEMORY连接的服务器。SHARED_MEMORY_GUI类似于直接显示，但将尝试连接到外部图形服务器以进行显示。子弹检查器浏览器有一个选项，以作为物理服务器或图形服务器。 |
| 可选择的 | 钥匙 | int | 在SHARED_MEMORY模式下，使用可选的共享内存密钥。当启动检查浏览器或SharedMemoryPhysics_*时，您可以使用可选的命令--共享_内存_SharedMemoryPhysics__键来设置键。这允许在同一台机器上运行多个服务器。 |
| 可选择的 | 主机名 (UDP和TCP) | 细绳 | IP地址或主机名，例如“127.0.0.1”或“本地主机”或“我的机器”。领域com” |
| 可选择的 | 港口 (UDP和TCP) | 整数 | 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相匹配） |
| 可选择的 | 选择 | 细绳 | 请将命令行选项传递到GUI服务器中。 您可以在[0..1]范围内设置背景颜色，如下所示： p.connect (p.GUI, 选项=”——背景_color_red=1——背景_color_蓝色=1——背景_color_绿色=1”) 其他选项包括： ——mouse_move_multiplier=0.400000（鼠标灵敏度）—— mouse_wheel_multiplier=0.400000（鼠标滚轮灵敏度）——宽度= <int>窗口宽度，单位为像素 </int>窗口宽度，单位为像素 ——高度=<int>为窗口的高度，以像素为单位。—— mp4=moviname.mp4（录制电影，需要ffmpeg）——mp4fps=<int>（用于电影录制，每秒设置帧数）。 |

如果没有连接，则返回一个物理客户端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，它将假设物理客户端id为0。除了GUI之外，您可以连接到多个不同的物理服务器。

例如：

枪弹。连接(枪弹。直接)

pybullet.连接(弹丸。GUI, 选项=“-opengl2”)

pybullet.连接(弹丸。共享内存, 1234)

pybullet.连接(弹丸。UDP, “192.168.0.1”)

枪弹。连接(枪弹。UDP, “localhost”, 1234)

枪弹。连接(枪弹。TCP, “localhost”, 6667)

使用共享内存连接

有一些物理服务器允许共享内存连接：App_SharedMemoryPhysics，App_SharedMemoryPhysics_GUI和弹头示例浏览器在实验/物理服务器下有一个例子，允许共享内存连接。这将允许您在一个单独的过程中执行物理模拟和渲染。

你还可以通过共享内存连接到App_SharedMemoryPhysics_VR，这是一个虚拟现实应用程序，支持头戴式显示器和6d的跟踪控制器，如HTC Vive和Oculus Rift。由于ValveOpenVRSDK只能在窗口下正常工作，App_SharedMemoryPhysics_VR只能在窗口下使用预制作（最好）或cmake构建。

使用UDP或TCP网络进行连接

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库来进行可靠的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使用克隆套接字库。当使用SSH隧道从防火墙后面的机器到机器人模拟时，这一点很有用。例如，你可以在Linux上使用PyBullet运行控制堆栈或机器学习，而在虚拟现实中使用HTC Vive或Rift在Windows上运行物理服务器。

另一个UDP应用程序是App_PhysicsServerSharedMemoryBridgeUDP应用程序，它充当到现有物理服务器的桥：您可以通过UDP连接到这个桥，并且桥使用共享内存连接到物理服务器：桥在客户机和服务器之间传递消息。同样地，也有一个TCP版本（用TCP替换UDP）。

还有一个GRPC客户端和服务端支持，默认情况下不启用它。你可以试试使用——enable_grpc选项预构建系统(参见项目符号3/预制作4)。

注意：目前，客户端和服务端都需要是32位或64位的构建！

tuble_客户端

如果您想并行地使用多个独立的模拟，您可以使用pybullet_utils.bullet_client . 的一个实例tuble_客户端。公告客户端(连接模式=项目符号。GUI，选项= ‘ ’)与项目实例具有相同的API。它将自动添加适当的物理sclientid到每个API调用。PyBullet健身房环境使用meton_客户端允许并行训练多个环境，请参见env_bases中的实现。[py. 另一个小例子展示了如何有两个独立的实例，每个实例都有自己的对象，请参见多个场景。py.](#)

切断

您可以使用连接调用返回的物理客户端Id（如果非负数）断开与物理服务器的连接。一个“直接”或“GUI”的物理服务器将会关闭。一个单独的（进程外）物理服务器将继续运行。另请参阅“重置模拟”，以删除所有项目。

断开的参数：

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

设置重力

默认情况下，不启用重力。设置重力，允许您设置所有对象的默认重力。设置重力输入参数为：（无返回值）

| | | | |
|------|-----------------|-----|-------------------------|
| 必须的 | graX | 使漂浮 | 沿着X世界轴的重力 |
| 必须的 | 砾石 | 使漂浮 | 沿Y世界轴的重力 |
| 必须的 | gravZ | 使漂浮 | 沿Z世界轴的重力 |
| 可选择的 | physicsClientId | int | 如果您连接到多个物理服务器，您可以选择哪一个。 |

loadURDF, loadSDF, loadMJCF

loadURDF将向物理服务器发送一个命令，以从一个通用的机器人描述文件（URDF）中加载一个物理模型。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，它是由柳树车库和开源机器人基金会（OSRF）创建的。许多机器人都有公共的URDF文件，你可以在这里找到一个描述和教程：<http://wiki.ros.org/urdf/Tutorials>

重要提示：大多数关节（滑块、旋转、连续）默认启用电机，防止自由运动。这类似于一个具有非常高摩擦谐波驱动器的机器人关节。您应该使用小头设置关节电机控制模式和目标设置。

setJointMotorControl2. 有关更多信息，请参见设置联合电机控制2API。

警告：默认情况下，PyBullet会缓存一些文件以加快加载速度。您可以使用设置物理工程参数启用文件缓存（启用文件缓存=0）。

loadURDF参数为：

| | | | |
|------|---------|------|--|
| 必须的 | 文件名 | 细绳 | 到物理服务器的文件系统上的URDF文件的相对路径或绝对路径。 |
| 可选择的 | 基本位置 | vec3 | 在世界空间坐标[X、Y、Z]的指定位置上创建对象的底部。请注意，此位置是位于URDF链路的位置。如果惯性系是非零的，这与质心位置不同。使用重置“基准位置 and 方向”可以设置质心的位置/方向。 |
| 可选择的 | 基底方向 | vec4 | 在指定方向将对象的基础创建为世界空间四元数[X、Y、Z、W]。参见中的注释 basePosition. |
| 可选择的 | 使用最大坐标 | int | 默认情况下，URDF文件中的关节是使用简化坐标方法创建的：关节使用羽毛石铰接体算法（ABA，Bt多Bullet 2.x）模拟。使用“最大坐标”选项将为每个连杆创建一个6个自由度的刚体，并且这些刚体之间的约束将用于建模关节。为没有链接/关节的身体启用最大坐标可以大大提高性能，比如机器人抬起的物体。 |
| 可选择的 | 使用固定的基础 | int | 强制加载对象的基础为静态的 |
| 可选择的 | 旗 | int | 以下标志可以使用位OR 组合： URDF_MERGE_FIXED_LINKS：这将从URDF文件中删除固定链接，并合并结果链接。这对性能很好，因为有各种算法 |

| | | | |
|------|------|-----|--|
| | | | <p>（铰接体算法、前向运动学等）在关节的数量上具有线性复杂度，包括固定关节。</p> <p>URDF_USE_INERTIA_FROM_FILE：默认情况下，子弹根据碰撞形状的质量和体积重新计算惯性张量。如果您能提供精确的惯性张量，请使用此标志。</p> <p>URDF_USE_SELF_COLLISION：默认情况下，子弹会禁用自碰撞。这个标志可以让你来启用它。您可以使用以下标志来自定义自碰撞行为：</p> <p>URDF_USE_SELF_COLLISION_INCLUDE_PARENT将启用子和父之间的冲突，默认情况下是禁用的。需要与URDF_USE_SELF_COLLISION标志一起使用。</p> <p>URDF_USE_SELF_COLLISION_EXCLUDE_ALL_PARENTS将放弃一个子链接和它的任何祖先（父母，父母的父母，直到基础）之间的自我碰撞。需要与URDF_USE_SELF_COLLISION一起使用。</p> <p>URDF_USE_IMPLICIT_CYLINDER，将使用一个平滑的隐式圆柱体。默认情况下，子弹会将把圆柱体镶嵌成一个凸包。</p> <p>URDF_ENABLE_SLEEPING，将允许在身体有一段时间没有移动后禁用模拟。与活动体的交互将重新启用模拟。</p> <p>URDF_INITIALIZE_SAT_FEATURES，将为凸面形状创建三角形网格。这将提高可视化效果，也允许使用分离轴测试（SAT），而不是GJK/EPA。要求使用设置的物理工程参数来启用sat。</p> <p>URDF_USE_MATERIAL_COLORS_FROM_MTL，将使用来自波前OBJ文件的RGB颜色，而不是来自URDF文件。</p> <p>URDF_ENABLE_CACHED_GRAPHICS_SHAPES，将缓存和重用图形形状。它将提高具有类似图形资产的文件的加载性能。</p> <p>URDF_MAINTAIN_LINK_ORDER，将尝试维护从URDF文件中获得的链接顺序。在URDF文件中，顺序是：辅助链接0、子链接1（附加到辅助链接0）、子链接2（附加到辅助链接0）。如果没有这个标志，顺序可以是P0，C2，C1。</p> |
| 可选择的 | 全局缩放 | 使漂浮 | 全局比例化将对URDF模型应用一个比例因子。 |

| | | | |
|------|-----------------|-----|----------------------|
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |
|------|-----------------|-----|----------------------|

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，而不是有效的主体唯一ID。

默认情况下，loadURDF将使用一个凸包来进行网格碰撞检测。对于静态（质量=0，不移动）网格，可以通过在URDF中添加标记使网格凹：

<链接凹= “yes” 名字= “baseLink”>看到武士。以urdf为例。URDF格式还有一些其他扩展，您可以浏览这些示例进行探索。PyBullet并不处理来自URDF文件的所有信息。请参阅示例和URDF文件，以了解支持哪些特性。通常会有一个Python API来控制该特性。每个链接只能有一个材质，所以如果你有多个不同材质的视觉形状，你需要将它们分割成单独的链接，由固定的关节连接。您可以使用OBJ2SDF实用程序来完成这一点，作为子弹的一部分。

loadSDF, loadMJCF

您还可以从其他文件格式中加载对象，例如。子弹sdf和。mjcf. 这些文件格式支持多个对象，因此返回值是对象唯一id的列表。SDF格式的详细说明见<http://sdformat.org>. loadSDF命令只提取了与机器人模型和几何形状相关的一些基本部分，而忽略了许多与摄像机、灯光等相关的元素。loadMJCF命令执行在OpenAI Gym中使用的MuJoCo MJCF xml文件的基本导入。另请参阅与默认关节电机设置相关的重要说明，并确保使用设置的关节电机控制2。

| | | | |
|------|-----------------|-----|---|
| 必须的 | 文件名 | 细绳 | 到物理服务器的文件系统上的URDF文件的相对路径或绝对路径。 |
| 可选择的 | 使用最大坐标 | int | 实验。更多细节请参见loadURDF。 |
| 可选择的 | 全局缩放 | 使漂浮 | SDF和URDF支持全局缩放，而不是MJCF。每个对象都将使用此比例因子（包括链接、链接框架、连接附件和线性连接限制）进行缩放。这对质量没有影响，只对几何形状有影响。如果需要，可以使用更改动力学来更改质量。 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

mJCF将返回一个对象唯一id数组：

| | | |
|-----------------|-------|-----------------------|
| objectUniqueIds | int列表 | 该列表包含已加载的每个对象的对象唯一id。 |
|-----------------|-------|-----------------------|

saveState, saveBullet, restoreState

当您在恢复到以前保存的状态后需要进行确定性模拟时，就需要存储所有重要的状态信息，包括接触点。保存世界的命令是不够的。您可以使用恢复状态命令从使用saveState（在内存中）或save子弹（在磁盘上）拍摄的快照中进行恢复。

保存状态命令只接受一个可选的客户端entServerId作为输入，并返回状态ID。保存弹头命令将该状态保存到a。磁盘上的项目符号文件。
恢复状态命令的输入参数为：

| | | | |
|------|----------------|-----|----------------------------|
| 可选择的 | 文件名 | 细绳 | 的文件名。使用保存的项目符号命令创建的项目符号文件。 |
| 可选择的 | stateId | int | 由保存状态返回的状态id |
| 可选择的 | clientServerId | int | 如果您已连接到多个服务器，则您可以选择一个服务器 |

文件名或状态id都需要有效。注意，恢复状态将将对象的位置和关节角度重置到保存状态，以及恢复接触点信息。在调用恢复状态之前，需要确保设置对象和约束。[请参见保存恢复状态。](#)py的例子。

删除状态

远程状态允许从内存中删除以前存储的状态。

拯救世界

您可以将当前世界的近似快照创建为Python文件，存储在服务器上。saveWorld可以作为一个基本的编辑功能，设置机器人、关节角度、虚拟位置对象位置和环境。稍后，您可以只需加载py子弹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，并非所有的设置都是这样的
存储在世界文件中。

输入参数为：

| | | | |
|-----|-----|----|-----------|
| 必须的 | 文件名 | 细绳 | 项目文件的文件名。 |
|-----|-----|----|-----------|

| | | | |
|------|----------------|-----|--------------------------|
| 可选择的 | clientServerId | int | 如果您已连接到多个服务器，则您可以选择一个服务器 |
|------|----------------|-----|--------------------------|

createCollisionShape/VisualShape

虽然世界上推荐的和最简单的方法是使用加载函数（loadURDF/SDF/MJCF/Bullet），你也可以通过编程方式创建碰撞和视觉形状，并使用它们使用创建多体。[查看创建的多身体链接.py](#)和[创建可视化形状](#)。子弹物理SDK中的Py例子。

创建碰撞形状的输入参数为

| | | | |
|------|------------------------|---------------|---|
| 必须的 | 形状类型 | int | geom_sphere , geom_box , geom_capsule , geom_cylinder , geom_plane , geom_mesh , GEOM_HEIGHTFIELD |
| 可选择的 | 半径 | 使漂浮 | 默认值0.5: GEOM_SPHERE、GEOM_CAPSULE、GEOM_CYLINDER |
| 可选择的 | halfExtents | vec3列表中的3个浮点数 | 默认[1, 1, 1]: 用于GEOM_BOX |
| 可选择的 | 高度 | 使漂浮 | 默认值: 1: 对于GEOM_CAPSULE, GEOM_CYLINDER |
| 可选择的 | 文件名 | 细绳 | 文件名，目前只有波前。obj. 将为.obj文件中的每个对象（标记为“o”）创建凸壳。 |
| 可选择的 | 网格比例 | vec3列表中的3个浮点数 | 默认值: [1, 1, 1], 用于GEOM_MESH |
| 可选择的 | 平面法线 | vec3列表中的3个浮点数 | 默认值: GEOM_PLANE的[0, 0, 1] |
| 可选择的 | 旗 | int | GEOM_FORCE_CONCAVE_TRIMESH: 对于GEOM_MESH, 这将创建一个凹形的静态三角形网格。这不应用于动态/移动对象，仅适用于静态（质量= 0）地形。 |
| 可选择的 | collisionFramePosition | vec3 | 碰撞形状相对于连杆框架的平移偏移量 |
| 可选择的 | 碰撞框架或orientation | vec4 | 碰撞形状相对于链接框架的旋转偏移量（四元数x、y、z、w） |
| 可选择的 | 至高点 | …的清单 vec3 | 高度域的定义。请参见 heightfield.py 的例子。 |
| 可选择的 | 指数 | int列表 | 高度场的定义 |
| 可选择的 | 高度字段文本缩放 | 使漂浮 | 高度场的纹理缩放 |

| | | | |
|------|------------------------------|-----|--------------------------------|
| 可选择的 | numHeightfield R ows | int | 高度场的定义 |
| 可选择的 | numHeightfield C olumns | int | 高度场的定义 |
| 可选择的 | replaceHeightf iel dIndex | int | 替换现有的高场（更新其高度）（比删除和重新创建高场要快得多） |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

返回值是碰撞形状的非负int唯一id，如果调用失败，则为-1。

创建碰撞形状数组

碰撞形状阵列是创造碰撞形状的阵列版本。有关用法，请参见蛇.py或创建可视化形状阵列。
关于如何使用它的一些例子。

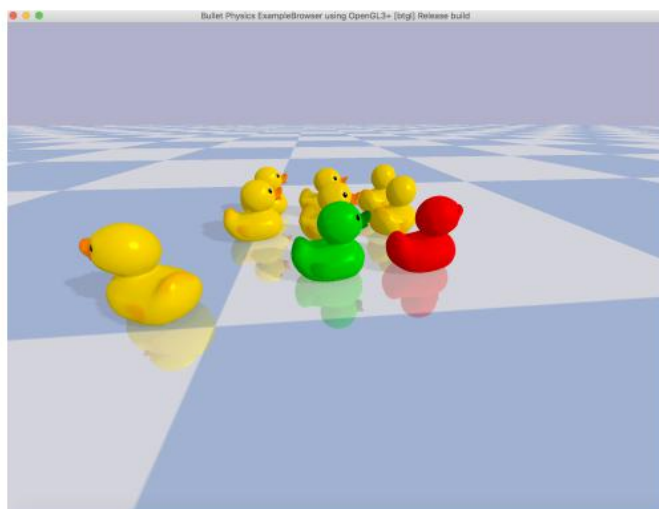
删除碰撞形状

远程碰撞形状将删除一个现有的碰撞形状，使用其碰撞形状的唯一id。

创建视觉形状

您可以以类似于创建碰撞形状的方式创建视觉形状，并添加一些附加参数来控制视觉外观，如漫反射颜色和镜面颜色。当您使用GEOM_MESH类型时，您可以指向一个波前OBJ文件，并且视觉形状将解析材料文件中的一些参数（。并加载一个纹理。请注意，较大的纹理（高于1024x1024像素）可以降低加载和运行时的性能。

请看到“examples/pybullet/examples/addPlanarReflection.” py和创建可视化形状.py



输入参数为

| | | | |
|------|---------------|-----------------|--|
| 必须的 | 形状类型 | int | geom_sphere , geom_box , geom_capsule , geom_cylinder , geom_plane , geom_mesh |
| 可选择的 | 半径 | 使漂浮 | 默认值0.5: 仅适用于GEOM_SPHERE, geom_capsule , geom_cylinder |
| 可选择的 | halfExtents | vec3列表中的3个浮点数 | 默认的[1, 1, 1]: 仅为GEOM_BOX设置 |
| 可选择的 | 长度 | 使漂浮 | 默认值: 1: 仅适用于GEOM_CAPSULE, GEOM_CYLINDER (长度=高度) |
| 可选择的 | 文件名 | 细绳 | 文件名, 目前只有波前.obj. 将为.obj文件中的每个对象 (标记为“o”) 创建凸壳。 |
| 可选择的 | 网格比例 | vec3列表中的3个浮点数 | 默认值: [1, 1, 1], 仅适用于GEOM_MESH |
| 可选择的 | 平面法线 | vec3列表中的3个浮点数 | 默认值: [0, 0, 1]仅适用于GEOM_PLANE |
| 可选择的 | 旗 | int | 待使用 |
| 可选择的 | rgbaColor | vec4, 列表4 彩车 | 红色、绿色、蓝色和alpha的颜色组件, 每个组件都在范围内[0..1]。 |
| 可选择的 | specularColor | vec3, 列表3 彩车 | 镜面反射颜色, 红色, 绿色, 范围 蓝色组件 |
| 可选择的 | 视觉框架位置 | vec3, 列表3 彩车 | 视觉形状相对于链接框架的平移偏移量 |
| 可选择的 | 至高点 | vec3列表 | 您可以提供顶点、索引、uv和法线, 而不是从obj文件中创建网格。 |
| 可选择的 | 指数 | int列表 | 三角形指数, 应该是3的倍数 |
| 可选择的 | uvs | vec2列表 | 为顶点的紫外线纹理坐标。使用 |

| | | | |
|------|------------------------------------|----------------|--------------------------------|
| | | | 改变视觉形状来选择纹理图像。uv的数量应该等于顶点的数量 |
| 可选择的 | 正常人 | vec3列表 | 顶点法线，数字应该等于顶点的数量。 |
| 可选择的 | visualFrameOrientati i 在...上 | vec4，列表4 彩车 | 相对于链接框架的视觉形状的旋转偏移量（四元数x、y、z、w） |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

返回值是视觉形状的非负int唯一id，如果调用失败，则为-1。[请参见创建可视化形状，createVisualShapeArray 并创建纹理网格视觉形状示例。](#)

创建可视化形状数组

创建可视化数列是创建可视化形状的数组版本。看createVisualShapeArray.py的例子。

创建多主体

虽然世界上最简单的创建方法是使用加载函数（loadURDF/SDF/MJCF/Bullet），但您可以使用创建多体创建一个多体。[请参阅创建多实体链接.py](#) 子弹物理SDK中的例子。它的生成参数与URDF和SDF参数非常相似。

您可以创建一个没有关节/子链接的多体，也可以创建一个带有关节/子链接的多体。如果您提供链接，确保每个列表的大小是相同的（链接质量）====（链接碰撞形状指数）等）。创建主体的输入参数为：

| | | | |
|------|----------|----------------|--------------------------------------|
| 可选择的 | 基本质量 | 使漂浮 | 底座的质量，以公斤为单位（如果使用SI单位） |
| 可选择的 | 基底碰撞形状指数 | int | 唯一的id从创造的碰撞形状或-1。您可以对多个多体重用碰撞形状（实例化） |
| 可选择的 | 基本视觉形状索引 | int | 唯一的id从创建可视化形状或-1。您可以重用视觉形状（实例化） |
| 可选择的 | 基本位置 | vec3, 3个浮点数的列表 | 笛卡尔世界的位置 |
| 可选择的 | 基底方向 | vec4, 4个浮点数的列表 | 四元数取向[x、y、z、w] |
| 可选择的 | 基座惯性框架位置 | vec3, 3个浮点数的列表 | 惯性系的局部位置 |

| | | | |
|------|-------------------------------|----------------|---|
| 可选择的 | 基座惯性框架方向 | vec4, 4个浮点数的列表 | 惯性系的局部方向 |
| 可选择的 | 链接质量 | 浮子列表 | 质量值的列表，每个链接对应一个。 |
| 可选择的 | 链接碰撞形状索引 | int列表 | 唯一id的列表，每个链接对应一个。 |
| 可选择的 | 链接视觉形状索引 | int列表 | 每个链接的视觉形状唯一id的列表 |
| 可选择的 | 链接位置 | vec3列表 | 本地链接位置的列表，关于父链接的位置 |
| 可选择的 | linkOrientations | vec4列表 | 本地链接方向的列表，w. r. t. 父母 |
| 可选择的 | 链路惯性帧位置 | vec3列表 | 局部惯性系pos的列表。在链接帧中 |
| 可选择的 | linkInertialFrameOrientations | vec4列表 | 局部惯性系角的列表。在链接帧中 |
| 可选择的 | 链接父级索引 | int列表 | 父链接的链接索引或基本链接的0。 |
| 可选择的 | 连杆接头类型 | int列表 | 关节类型的列表，每个链接对应一个。目前还支持JOINT_REVOLUTE、JOINT_PRISMATIC、JOINT_SPHERICAL和JOINT_FIXED类型。 |
| 可选择的 | 连杆连接轴 | vec3列表 | 局部框架中的关节轴 |
| 可选择的 | 使用最大坐标 | int | 实验性的，最好是0/假的。 |
| 可选择的 | 旗 | int | 类似于在loadURDF中传递的标志，例如URDF_USE_SELF_COLLISION。请参见loadURDF中的标志解释。 |
| 可选择的 | 批处理位置 | vec3列表 | 基本位置数组，用于快速批量创建许多多体。 请参见示例。 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

creaty的返回值为非负唯一id或-1。样例

```
cuid = pybullet.createCollisionShape (pybullet.GEOM_BOX, 半范围= [1,1,1])
质量=0#静态盒
pybullet.createMultiBody (质量, 角)
参见创建多实体链接.py, createObstacleCourse.py和创建可视化形状。在子弹/子弹文件夹
中的py。
```

获取网格数据

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

| | | | |
|-----|--------------|-----|--------|
| 必须的 | bodyUniqueId | int | 实体唯一ID |
|-----|--------------|-----|--------|

| | | | |
|------|-----------------|-----|--|
| 可选择的 | 链接索引 | int | 环比指数 |
| 可选择的 | 碰撞形状索引 | int | 复合形状索引，如果链接中存在多个碰撞形状（参见碰撞数据） |
| 可选择的 | 旗 | int | 默认情况下，PyBullet将返回图形渲染顶点。由于具有不同法线的顶点是重复的，因此可以比原始网格中的顶点更多。您可以通过使用标志=来接收模拟顶点 pybullet.mesh_data_模拟网格 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

stepSimulation, performCollisionDetection

逐步仿真将在一个单一的正向动力学仿真步骤中执行所有的动作，如碰撞检测、约束求解和集成。默认的时间步长为1/240秒，它可以使用设置的时间步长或设置的物理工程参数API进行更改。

步骤模拟输入参数是可选参数：

| | | | |
|------|-----------------|-----|----------------------|
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |
|------|-----------------|-----|----------------------|

默认情况下，步骤模拟没有返回值。

仅用于实验/高级使用：如果通过设置物理工程参数API启用了报告溶解分析，以下信息将作为岛屿信息列表返回，详细信息如下：

| | | |
|-------------------|------------|---------------|
| islandId | int | 岛独特的id |
| numBodies | 主体唯一性id的列表 | 这个岛上的身体是独特的身份 |
| numIterationsUsed | int | 解算器的数量使用的迭代。 |
| 剩余剩余 | 使漂浮 | 剩余约束错误。 |

参见设置实时模拟，以自动让物理服务器运行基于其实时时钟的动态模拟。

执行碰撞检测

执行碰撞检测的api将只执行逐步模拟的碰撞检测阶段。唯一的参数是物理客户端id。在拨打这个电话后，您可以使用获取联系人点。

设置实时仿真

默认情况下，物理服务器将不会逐步模拟，除非您显式地发送一个“逐步模拟”命令。这样一来，您就可以维护模拟的控制决定论。通过让物理服务器使用实时模拟命令，让物理服务器根据其实时时钟（RTC）自动进行模拟，可以实时运行模拟。如果您启用了实时模拟，则不需要调用“逐步模拟”。

注意，在直接时间模拟模式下没有效果：在直接模式下，物理服务器和客户端发生在同一个线程中，你触发每个命令。在GUI模式和虚拟现实模式以及TCP/UDP模式下，物理服务器与客户端单独的线程（PyBullet）运行，relt时间模拟允许物理服务器线程向逐步模拟添加额外的调用。

输入参数为：

| | | | |
|------|-----------------|-----|----------------------|
| 必须的 | 启用实时仿真 | int | 0来禁用实时模拟，1来启用 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

获取基本位置和方向

获取“基础位置和方向”报告笛卡尔世界坐标中主体的基础（或根链接）的当前位置和方向。方向是一个以[xyzw]格式的四元数。

获取基准位置和方向”的输入参数为：

| | | | |
|------|-----------------|-----|----------------------|
| 必须的 | objectUniqueId | int | 对象唯一id，从loadURDF返回。 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

获取基本位置和方向返回3个浮点的位置列表和以[x、y、z、w]顺序返回4个浮点的的方向的列表。如果需要，使用得到欧拉的四元数将四元数转换为欧拉。

另请参见重置基准位置和方向，以重置对象的位置和方向。

这就完成了第一个PyBullet脚本。子弹附带了子弹/数据文件夹中的几个URDF文件。

重置基础位置和方向

您可以重置每个对象的基部（根部）的位置和方向。最好只在开始时这样做，而不是在运行模拟期间，因为该命令将覆盖所有物理模拟的效果。线性速度和角速度被设为零。可以使用“重置基准速度”重置为非零线速度和/或角速度。

要重置基准位置和方向的输入参数为：

| | | | |
|------|-----------------|------|--------------------------------|
| 必须的 | bodyUniqueId | int | 对象唯一id，从loadURDF返回。 |
| 必须的 | posObj | vec3 | 在世界空间坐标[X、Y、Z]中指定位置的底部 |
| 必须的 | ornObj | vec4 | 在指定方向将对象的底部重置为世界空间四元数[X、Y、Z、W] |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

没有返回参数。

变换：位置 and 方向

物体的位置可以用笛卡尔世界空间坐标[x, y, z]来表示。物体的方向（或旋转）可以用四元数、x、y、z、w]、欧拉角、偏航、俯仰、滚动、]或3x3矩阵来表示。PyBullet提供了一些辅助函数之间的转换四元数，欧拉角和3x3矩阵。此外，还有一些函数来进行相乘变换和反转变换。

小子弹API使用四元数来表示方向。由于四元数对人来说不是很直观，所以有两个api可以在四元数和欧拉角之间进行转换。它的输入参数为：

| | | | |
|------|-----------------|------------------|--|
| 必须的 | 欧拉角 | vec3: 3的列表 彩车 | X、Y、Z欧拉角以弧度表示，累积3个旋转，表示绕X的滚动，绕Y的俯仰，绕Z轴的偏转。 |
| 可选择的 | physicsClientId | int | 未使用的，添加的API一致性。 |

返回一个四元数，vec4, 4个浮点值列表[X, Y, Z, W]。

getEulerFromQuaternion

getEulerFrom四元数输入参数为:

| | | | |
|------|-----------------|----------------|------------------|
| 必须的 | 四个一组 | vec4: 4个浮点数的列表 | 四元数的格式是[x、y、z、w] |
| 可选择的 | physicsClientId | int | 未使用的，添加的API一致性。 |

返回一个包含3个浮点值的列表，一个vec3。旋转顺序是首先在X周围滚动，然后在Y周围俯仰，最后在Z周围偏航，就像ROS URDF的惯例一样。

getMatrixFromQuaternion

这是一个实用程序API，用于从一个四元数中创建一个3x3的矩阵。输入是一个四元数，并输出一个包含9个浮点数的列表，表示该矩阵。

从四元数中获得轴角

从四元数得到轴角将返回一个给定的四元数方向的轴和角度表示。

| | | | |
|------|-----------------|--------|-----------------|
| 必须的 | 四个一组 | 4个浮子列表 | 方向 |
| 可选择的 | physicsClientId | int | 未使用的，添加的API一致性。 |

multiplyTransforms, invertTransform

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

多重变换的输入参数为:

| | | | |
|------|-----------------|----------------|-----------------|
| 必须的 | 位置A | vec3, 3个浮点数的列表 | |
| 必须的 | 方向A | vec4, 4个浮点数的列表 | 四元数[x, y, z, w] |
| 必须的 | positionB | vec3, 3个浮点数的列表 | |
| 必须的 | orientationB | vec4, 4个浮点数的列表 | 四元数[x, y, z, w] |
| 可选择的 | physicsClientId | int | 未使用的，添加的API一致性。 |

返回值是位置（vec3）和方向（vec4、四元数x、y、x、w）的列表。逆变变换的输入

和输出参数为：

| | | | |
|-----|----|----------------|-----------------|
| 必须的 | 位置 | vec3, 3个浮点数的列表 | |
| 必须的 | 方向 | vec4, 4个浮点数的列表 | 四元数[x, y, z, w] |

逆变变换的输出是一个位置（vec3）和方向（vec4，四元数x、y、x、w）。

getDifferenceQuaternion

四元数将返回一个从开始方向到结束方向的四元数。

| | | | |
|------|-----------------|--------|-----------------|
| 必须的 | quaternionStart | 4个浮子列表 | 开始方向 |
| 必须的 | quaternionEnd | 4个浮子列表 | 末端方向 |
| 可选择的 | physicsClientId | int | 未使用的，添加的API一致性。 |

获取API版本

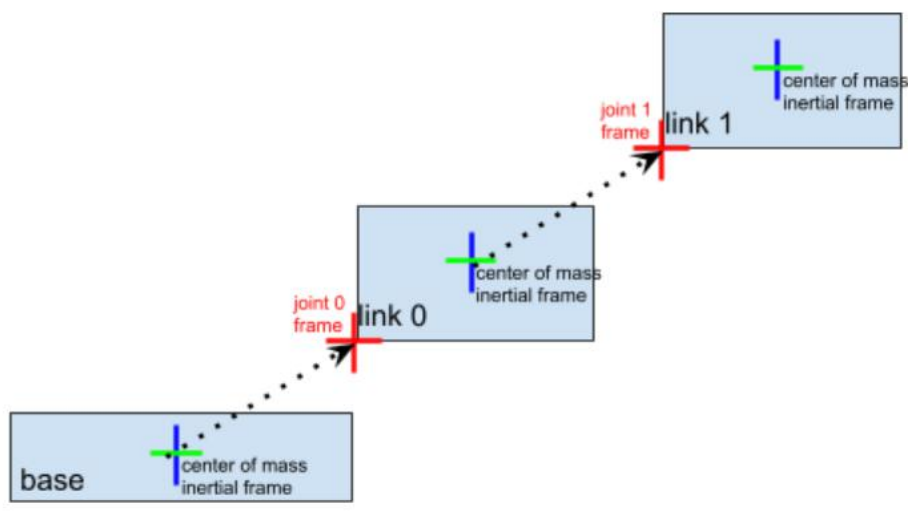
您可以以年-月-0-天的格式查询API版本。您只能在相同API版本的物理客户端/服务器之间进行连接，并与相同的位数（32位/64位）之间进行连接。有一个可选的未使用的参数物理clientid，是为API一致性而添加的。

| | | | |
|------|-----------------|-----|-----------------|
| 可选择的 | physicsClientId | int | 未使用的，添加的API一致性。 |
|------|-----------------|-----|-----------------|

控制机器人

在介绍中，我们已经展示了如何初始化PyBullet和加载一些对象。如果您将loadURDF命令中的文件名替换为“r2d2”。“你可以从ROS教程中模拟一个R2D2机器人。让我们控制这个R2D2机器人来移动，四处看看和控制抓握器。为此，我们需要知道如何访问其联合电机。

底座、接头、连接



在URDF文件中描述的模拟机器人有一个基础，可选择通过关节连接连接。每个关节将一个父链接连接到一个子链接。在层次结构的根有一个根父，我们称之为父。底座可以是完全固定的，0个自由度，或完全自由的，有6个自由度。由于每个链路都通过单个关节连接到父节点，因此关节的数量等于链路的数量。常规链接的链接索引在“[0”范围内。（）]由于基础不是常规的‘链接’，我们使用-1的约定作为链接索引，但是大多数与链接相关的api只接受链接索引 ≥ 0 的常规链接。我们使用联合坐标系相对于母质心惯性坐标系表示的约定，它与主惯性轴对齐。

getNumJoints, getJointInfo

加载机器人后，您可以使用getNumJoints API查询关节的数量。对于r2d2。这应该返回15。

getNumJoints输入参数：

| | | | |
|-----|-----------------|-----|----------------------|
| 必须的 | bodyUniqueId | int | 主体唯一id，由加载URDF等返回。 |
| 可选的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

getNumJoints返回一个表示关节数的整数值。

getJointInfo

对于每个关节，我们可以查询一些信息，如其名称和类型。

获取Joint信息输入参数

| | | | |
|-----|-----------------|-----|--------------------------------------|
| 必须的 | bodyUniqueId | int | 主体唯一id，由加载URDF等返回。 |
| 必须的 | 联合指数 | int | ..范围内的索引[0getNumJoint（bodyuniqueId）) |
| 可选的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

getJointInfo返回一个信息列表：

| | | |
|--------------|------|---|
| 联合指数 | int | 与输入参数相同的联合索引 |
| 联合名称 | 细绳 | 在URDF（或SDF等）文件中指定的接头的名称 |
| 接头类型 | int | 关节的类型，这也意味着位置和速度变量的数量。关节旋转，关节棱柱形，关节球面，关节平面，关节固定。有关更多细节，请参见基础、关节和链接上的部分。 |
| qIndex | int | 此主体的位置状态变量中的第一个位置索引 |
| uIndex | int | 在这个物体的速度状态变量中的第一个速度指数 |
| 旗 | int | 预订的 |
| jointDamping | 使漂浮 | 在URDF文件中指定的关节阻尼值 |
| 关节摩擦 | 使漂浮 | 在URDF文件中指定的关节摩擦值 |
| 接头下限 | 使漂浮 | 滑块和旋转（铰链）接头的位置下限，如URDF文件中规定。 |
| 接头上限 | 使漂浮 | 滑块和旋转关节的位置上限。在上限<下限时忽略值。 |
| 联合最大力 | 使漂浮 | 在URDF（可能是其他文件格式中指定的最大力）请注意，这个值不会自动使用。你可以在“设置关节运动控制2”中使用最大力。 |
| 关节最大速度 | 使漂浮 | 在URDF中规定的最大速度。注意，目前实际电机控制命令中没有使用最大速度。 |
| 链接名称 | 细绳 | 在URDF（或SDF等）中指定的链接的名称。文件 |
| 关节轴 | vec3 | 局部帧中的联合轴（忽略JOINT_FIXED） |

| | | |
|----------------|------|-----------------------|
| parentFramePos | vec3 | 母框架中的接头位置 |
| parentFrameOrn | vec4 | 父框架中的关节取向（四元数x、y、z、w） |
| 父索引 | int | 父链接索引，-1表示基数 |

setJointMotorControl2/Array

注：设置联合汽车控制已经过时，并由设置联合汽车控制2API所取代。（或者更好的是使用设置联合汽车控制阵列）。

我们可以通过为一个或多个关节电机设置一个所需的控制模式来控制一个机器人。在逐步模拟过程中，物理引擎将模拟电机，以达到给定的目标值，即在最大电机力和其他约束条件范围内可以达到的目标值。

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。您可以通过使用最大力为0来禁用这些默认电机。这将让您执行扭矩控制。
例如：

```
maxForce = 0
模式=p.速度控制
p.setJointMotorControl2 (objUid, jointIndex,
    controlMode=mode, force=maxForce)
```

你也可以使用一个小的非零力来模拟关节摩擦。

如果你想要一个轮子保持恒定的速度，你可以使用最大的力：

```
maxForce = 500
p.setJointMotorControl2 (bodyUniqueId=objUid,
    jointIndex=0,
    controlMode=p.速度控制,
    targetVelocity = targetVel,
    力=最大力)
```

设置联合电机控制2的输入参数为：

| | | | |
|-----|--------------|-----|--|
| 必须的 | bodyUniqueId | int | 主体唯一的id作为从加载URDF等返回。 |
| 必须的 | 联合指数 | int | 链接索引在范围内[0.. (budiqueId)（注意链接索引==关节索引） |
| 必须的 | 控制模式 | int | POSITION_CONTROL(事实上是 control_mode_position_velocity_pd), VELOCITY_CONTROL, TORQUE_CONTROL和 |

| | | | |
|------|-----------------|-----|--|
| | | | pd_control . (也有稳定 (隐式) PD控制的实验 STABLE_PD_CONTROL, 这需要额外的准备。看到人类运动捕获。py和pybullet_envs。deep_mimc为STABLE_PD_CONTROL的示例。)TORQUE_CONTROL将立即应用一个扭矩, 因此它只有在显式地调用逐步模拟时才有效。 |
| 可选择的 | 目标位置 | 使漂浮 | 在POSITION_CONTROL中, 目标值是关节的目标位置 |
| 可选择的 | 目标速度 | 使漂浮 | 在VELOCITY_CONTROL和POSITION_CONTROL中, 目标速度是关节的期望速度, 请参见下面的实现说明。请注意, 目标速度并不是最大关节速度。在PD_CONTROL和POSITION_CONTROL/CONTROL_MODE_POSITION_VELOCITY_PD中, 最终目标速度的计算方法是使用: $kp*(erp*(desiredPosition - currentPosition)/dt) + currentVelocity + kd*(m_desiredVelocity - currentVelocity)$ 另请参见 examples/pybullet/examples/pdControl.py |
| 可选择的 | 强迫 | 使漂浮 | 在POSITION_CONTROL和VELOCITY_CONTROL中, 这是用于达到目标值的最大运动力。在TORQUE_CONTROL中, 这是每个模拟步骤所要施加的力/扭矩。 |
| 可选择的 | 位置增益 | 使漂浮 | 参见下面的实施说明 |
| 可选择的 | 速度增益 | 使漂浮 | 参见下面的实施说明 |
| 可选择的 | maxVelocity | 使漂浮 | 在POSITION_CONTROL中, 这将速度限制在最大值 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器, 您可以选择一台。 |

注: 联合电机控制器的实际实现是作为POSITION_CONTROL和VELOCITY_CONTROL的一个约束条件, 并作为一个外力
 扭矩控制:

| 方法 | 实施 | 组成部分 | 要最小化的约束误差 |
|------|----|---------|---|
| 位置控制 | 限制 | 速度和位置限制 | 错误= 位置增益* (期望位置- actual_position) + 速度增益* (目标速度-实际速度) |
| 速度控制 | 限制 | 纯速度约束 | 误差=期望速度-实际速度 |
| 扭矩控制 | 外力 | | |

通常, 最好从VELOCITY_CONTROL或POSITION_CONTROL开始。要做TORQUE_CONTROL (力控制) 要困难得多, 因为模拟正确的力依赖于非常精确的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)。

设置联合电机控制阵列

您可以为所有输入传递数组，以显著减少调用开销，而不是为每个关节进行单独的调用。

除了用整数列表替换整数外，关节控制阵列相同的参数采用与关节控制2相同的参数。

设置联合电机控制阵列的输入参数为：

| | | | |
|------|-----------------|-------|---|
| 必须的 | bodyUniqueId | int | 主体唯一的id作为从加载URDF等返回。 |
| 必须的 | 联合指数 | int列表 | 指数范围[0.. (budiqueId)（注意链接索引==关节索引） |
| 必须的 | 控制模式 | int | 位置控制，速度控制，扭矩控制，PD_CONTROL。（也有稳定（隐式）PD控制的实验STABLE_PD_CONTROL，这需要额外的准备。请参见《人类的动作捕捉》。py和pybullet_envs.deep_mimc为STABLE_PD_CONTROL的示例。） |
| 可选择的 | 目标位置 | 浮子列表 | 在POSITION_CONTROL中，目标值是关节的目标位置 |
| 可选择的 | 目标速度 | 浮子列表 | 在PD_CONTROL、VELOCITY_CONTROL和POSITION_CONTROL中，目标值是关节的目标速度，请参见下面的实现说明。 |
| 可选择的 | 力 | 浮子列表 | 在PD_CONTROL、POSITION_CONTROL和VELOCITY_CONTROL中，这是用于达到目标值的最大运动力。在TORQUE_CONTROL中，这是每个模拟步骤所要施加的力/扭矩。 |
| 可选择的 | 位置增益 | 浮子列表 | 参见下面的实施说明 |
| 可选择的 | 速度增益 | 浮子列表 | 参见下面的实施说明 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

请看到“bullet3/examples/pybullet/tensorflow/humanoid_running.” 作为一个使用设置联合汽车控制阵列的例子。

setJointMotorControlMultiDof

setJointMotorControlMultiDof类似于设置关节运动控制2，但它支持球形（多dof）关节，这是用于深度模拟环境（在pybullet_envs中）和人道运动捕捉。py的例子。而不是一个单一的浮子，目标位置，目标速度和
强制参数接受1个浮点列表或3个浮点列表，以支持球面关节。

setJointMotorControlMultiDof的输入参数是：

| | | | |
|------|-----------------|-------------|--|
| 必须的 | bodyUniqueId | int | 主体唯一的id作为从加载URDF等返回。 |
| 必须的 | 联合指数 | int | 链接索引在范围内[0.. (budiqueId) (注意链接索引==关节索引) |
| 必须的 | 控制模式 | int | POSITION_CONTROL (实际上是CONTROL_MODE_POSITION_VELOCITY_PD)、VELOCITY_CONTROL、TORQUE_CONTROL和PD_CONTROL。(也有稳定(隐式)PD控制的实验STABLE_PD_CONTROL, 这需要额外的准备。请参见humanoidMotionCapture.py和pybullet_envs.deep_mimc为STABLE_PD_CONTROL的示例。) |
| 可选择的 | 目标位置 | 1个或3个浮点数的列表 | 在POSITION_CONTROL中, 目标值是关节的目标位置 |
| 可选择的 | 目标速度 | 1个或3个浮点数的列表 | 在VELOCITY_CONTROL和POSITION_CONTROL中, 目标速度是关节的期望速度, 请参见下面的实现说明。请注意, 目标速度并不是最大关节速度。在PD_CONTROL和POSITION_CONTROL/CONTROL_MODE_POSITION_VELOCITY_PD中, 最终目标速度的计算方法是使用: $kp * (erp * (desiredPosition - currentPosition) / dt) + currentVelocity + kd * (m_desiredVelocity - currentVelocity)$. 另请参见examples/pybullet/examples/pdControl.py |
| 可选择的 | 强迫 | 1个或3个浮点数的列表 | 在POSITION_CONTROL和VELOCITY_CONTROL中, 这是用于达到目标值的最大运动力。在TORQUE_CONTROL中, 这是每个模拟步骤所要施加的力/扭矩。 |
| 可选择的 | 位置增益 | 使漂浮 | 参见下面的实施说明 |
| 可选择的 | 速度增益 | 使漂浮 | 参见下面的实施说明 |
| 可选择的 | maxVelocity | 使漂浮 | 在POSITION_CONTROL中, 这将速度限制在最大值 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器, 您可以选择一台。 |

setJointMotorControlMultiDofArray

setJointMotorControlMultiDofArray是在多个控制目标中传递的一个更有效的setJointMotorControlMultiDof, 版本, 以避免/减少Python和PyBullet C++扩展之间的调用开销。例子请参见humanoidMotionCapture.py。

setJointMotorControlMultiDofArray的输入参数是:

| | | | |
|-----|--------------|---------------|---|
| 必须的 | bodyUniqueId | int | 主体唯一的id作为从加载URDF等返回。 |
| 必须的 | 联合指数 | ...的清单 int | 链接索引在范围内[0.. (budiqueId) (注意链接索引==关节索引) |

| | | | |
|------|-----------------|---------------------|---|
| 必须的 | 控制模式 | int | POSITION_CONTROL（实际上是CONTROL_MODE_POSITION_VELOCITY_PD）、VELOCITY_CONTROL、TORQUE_CONTROL和PD_CONTROL。（也有稳定（隐式）PD控制的实验STABLE_PD_CONTROL，这需要额外的准备。请参见humanoidMotionCapture.py和pybullet_envs.deep_mimc为STABLE_PD_CONTROL的示例。） |
| 可选择的 | 目标位置 | …的清单 浮动或 vec3 | 在POSITION_CONTROL中，目标值是关节的目标位置 |
| 可选择的 | 目标速度 | …的清单 使漂浮 | 在VELOCITY_CONTROL和POSITION_CONTROL中，目标速度是关节的期望速度，请参见下面的实现说明。请注意，目标速度并不是最大关节速度。在PD_CONTROL和POSITION_CONTROL/CONTROL_MODE_POSITION_VELOCITY_PD中，最终目标速度的计算方法是使用： $kp*(erp*(desiredPosition - currentPosition)/dt) + currentVelocity + kd*(m_desiredVelocity - currentVelocity)$ 。另请参见examples/pybullet/examples/pdControl.py |
| 可选择的 | 力 | …的清单 使漂浮 | 在POSITION_CONTROL和VELOCITY_CONTROL中，这是用于达到目标值的最大运动力。在TORQUE_CONTROL中，这是每个模拟步骤所要施加的力/扭矩。 |
| 可选择的 | 位置增益 | …的清单 使漂浮 | 参见设置联合电机控制2中的实施说明 |
| 可选择的 | 速度增益 | …的清单 使漂浮 | 参见设置联合电机控制2中的实施说明 |
| 可选择的 | maxVelocities | …的清单 使漂浮 | 在POSITION_CONTROL中，这将速度限制在最大值 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

getJointState (s), resetJointState

我们可以使用关节状态从关节中查询几个状态变量，如关节位置、速度、关节反作用力和关节电机转矩。

获取Joint状态输入参数

| | | | |
|------|-----------------|-----|--|
| 必须的 | bodyUniqueId | int | 主体唯一id由加载URDF等返回 |
| 必须的 | 联合指数 | int | 链接索引在范围内[0..getNumJoints (bodyUniqueId)] |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

获取Joint状态输出

| | | |
|----------|--------|--|
| 联合位置 | 使漂浮 | 该接头的位置值。 |
| 关节速度 | 使漂浮 | 这个接头的速度值。 |
| 联合反应部队 | 6个浮子列表 | 这些是关节反作用力，如果为此接头启用了扭矩传感器，则它是[Fx、Fy、Fz、Mx、My、Mz]。没有扭矩传感器，为[0, 0, 0, 0, 0, 0]。 |
| 应用接头电机扭矩 | 使漂浮 | 这是在最后一步模拟中应用的电机转矩。请注意，这只适用于VELOCITY_CONTROL和POSITION_CONTROL。如果您使用TORQUE_CONTROL，那么所应用的关节电机扭矩正是您所提供的，所以不需要单独报告它。 |

获得联合国家

获取连接状态是获取连接状态的数组版本。不要传入单个连接索引，而是传入一个连接索引列表。

getJointStateMultiDof

还有球形关节。

获取Joint状态输出

| | | |
|----------|-------------------|--|
| 联合位置 | 1个或4个的列表 使漂浮 | 该关节的位置值（如jonit角/位置或关节方向四元数） |
| 关节速度 | 1个或3个之间的列表 使漂浮 | 这个接头的速度值。 |
| 联合反应部队 | 6个浮子列表 | 这些是关节反作用力，如果为此接头启用了扭矩传感器，则它是[Fx、Fy、Fz、Mx、My、Mz]。没有扭矩传感器，为[0, 0, 0, 0, 0, 0]。 |
| 应用接头电机扭矩 | 使漂浮 | 这是在最后一步模拟中应用的电机转矩。请注意，这只适用于VELOCITY_CONTROL和POSITION_CONTROL。如果您使用TORQUE_CONTROL，那么所应用的关节电机扭矩正是您所提供的，所以不需要单独报告它。 |

getJointStatesMultiDof

允许查询多个关节状态，包括多节点（球形）关节。

重置联合状态

您可以重置关节的状态。最好只在一开始就这样做，而不运行模拟：重置连接状态覆盖所有的物理模拟。请注意，我们目前只支持1-DOF的电动关节，滑动关节或旋转关节。

| | | | |
|------|-----------------|-----|---------------------------------|
| 必须的 | bodyUniqueId | int | 主体唯一id由加载URDF等返回 |
| 必须的 | 联合指数 | int | 范围内的关节指数[0..getNumJoints（车身单位）] |
| 必须的 | 目标值 | 使漂浮 | 接头位置（半径中的角度或位置） |
| 可选择的 | 目标速度 | 使漂浮 | 关节速度（角速度或直线速度） |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

resetJointState (s)MultiDof

对于球形关节也有重新连接状态。[请参阅人类运动捕获](#) 一个重置状态的例子。还要重置关节状态，以便一次重置多个关节。

启用联合力矩传感器

您可以在每个关节中启用或禁用关节力/扭矩传感器。一旦启用了，如果您执行一个逐步模拟，“getJointState”将报告固定自由度下的关节反作用力：一个固定的关节将测量所有6个自由度关节力/力矩。旋转铰链接头力/扭矩传感器将测量除铰链轴以外的5自由度反作用力。由关节电机所施加的力可在所应用的关节电机转矩中得到。

启用联合力力矩传感器的输入参数为：

| | | | |
|------|-----------------|-----|---------------------------------|
| 必须的 | bodyUniqueId | int | 主体唯一id由加载URDF等返回 |
| 必须的 | 联合指数 | int | 范围内的关节指数[0..getNumJoints（车身单位）] |
| 可选择的 | 启用传感器 | int | 1/True以启用，0/False禁用力/扭矩传感器 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

getLinkState (s)

您还可以使用getLinkState查询每个链接的质心的笛卡尔世界的位置和方向。它还将把质心的局部惯性系报告到URDF链接系，使计算图形/可视化系更容易。

获取链接状态输入参数

| | | | |
|------|--------------------------|-----|---------------------------------------|
| 必须的 | bodyUniqueId | int | 主体唯一id由加载URDF等返回 |
| 必须的 | 链接索引 | int | 环比指数 |
| 可选择的 | 计算链路速度 | int | 如果设置为1，则将计算并返回笛卡尔世界速度。 |
| 可选择的 | computeForwardKinematics | int | 如果设置为1（或True），笛卡尔世界将使用正向运动学重新计算位置/方向。 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

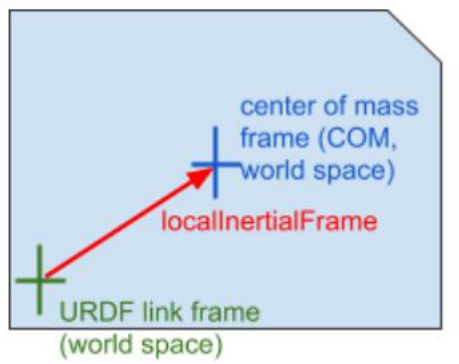
获取链接状态返回值

| | | |
|----------|----------------|---------------------------------------|
| 链接世界位置 | vec3, 3个浮点数的列表 | 质心的笛卡尔位置 |
| 链接世界定位 | vec4, 4个浮点数的列表 | 质心的笛卡尔方向，以四元数[x、y、z、w] |
| 局部惯性帧位置 | vec3, 3个浮点数的列表 | 用URDF链系表示的惯性系（质心）的局部位置偏移 |
| 局部惯性帧方向 | vec4, 4个浮点数的列表 | 以URDF连杆系表示的惯性系的局部方向（四元数[x、y、z、w]）偏移量。 |
| 世界链接框架位置 | vec3, 3个浮点数的列表 | URDF链接框架的世界位置 |
| 世界链接框架方向 | vec4, 4个浮点数的列表 | URDF链接框架的世界方向 |
| 世界链接线性速度 | vec3, 3个浮点数的列表 | 笛卡尔世界速度。只有在计算链接速度非零时才返回。 |
| 世界连接角速度 | vec3, 3个浮点数的列表 | 笛卡尔世界速度。只有在计算链接速度非零时才返回。 |

URDF链接框架和质心框架（都在世界空间中）之间的关系是： $\text{=通信链接框架} * \text{局部惯性框架}$ 。
相反的有关链路和惯性系的更多信息，请参见ROS [URDF 个别辅导时间](#)

获取链接状态

获取链接状态将返回多个链接的信息。它将接受链接索引作为一个链接的列表，而不是链接索引。这可以通过减少调用到获取LinkState的多次调用的开销来提高性能。



示例脚本（可能已经过时，请检查实际的子弹/示例/项目符号/示例文件夹。）

| | |
|--|--|
| 例如，压力流，人形机器人。py | 加载一个类人动物，并使用一个训练过的神经网络来控制运行，使用张量流，由OpenAI训练 |
| examples/pybullet/gym/pybullet_envs/bullet/minitaur.py和minitaur_gym_env.py | 你也可以使用python -m pybullet_envs. 例子minitaur_gym_env_example在你pip安装小子弹看到迷你龙在行动。 |
| examples/pybullet/examples/quadruped.py | 从URDF文件中加载一个四足动物，步骤模拟，控制电机为一个基于正弦波的简单跳跃步态。也将使用p记录状态到文件。startStateLogging. 请参阅视频。 |
| 示例/四倍播放.py | 创建一个四足动物（迷你龙），读取日志文件，并设置位置作为电机控制目标。 |
| examples/pybullet/examples/testrender.py | 加载一个URDF文件并渲染一个图像，获得像素（RGB，深度，分割掩码）和显示图像使用Matplotlib。 |
| examples/pybullet/examples/testrender_np.py | 类似于testrender.py，但使用NumPy数组加快像素传输。还包括简单的基准测试/计时。 |
| examples/pybullet/examples/saveWorld.py | 将对象的状态（位置、方向）保存到一个项目符号Python脚本中。这主要用于在VR中设置场景和保存初始状态。并非所有的状态都被序列化了。 |
| 例子，反运动学的例子.py | 展示如何使用计算逆运动学命令，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 | 以编程方式创建链接之间的约束。 |
| examples/pybullet/examples/vrhand.py | 使用VR手套控制一只手，由VR控制器跟踪。 请参阅视频。 |

getBaseVelocity, resetBaseVelocity

使用基准速度获取物体底部的线性速度和角速度。输入参数为：

| | | | |
|------|-----------------|-----|----------------------|
| 必须的 | bodyUniqueId | int | 主体唯一id，如从load*方法返回的。 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

这返回两个向量3值的列表，该列表表示笛卡尔世界空间坐标中的线速度[x、y、z]和角速度[w_x、w_y、w_z]。

您可以使用重置基准速度来重置车身底部的线性速度和/或角速度。输入参数为：

| | | | |
|------|-----------------|----------------|----------------------|
| 必须的 | objectUniqueId | int | 主体唯一id，如从load*方法返回的。 |
| 可选择的 | 线性速度 | vec3, 3个浮点数的列表 | 在笛卡尔世界坐标中的线性速度。 |
| 可选择的 | 角速度 | vec3, 3个浮点数的列表 | 笛卡尔世界坐标中的角速度。 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

applyExternalForce/Torque

您可以使用应用外部力和应用外部力矩对车身施加一个力或扭矩。注意，这种方法只在显式步进模拟时有效，换句话说：设置实际时间模拟(0)。在每个模拟步骤之后，外力被清除为零。如果您正在使用“设置Real时间模拟”(1)，则应用外力/扭矩将具有未定义的行为（0、1或多个力/扭矩应用）。

输入参数为：

| | | | |
|-----|----------------|----------------|--|
| 必须的 | objectUniqueId | int | 由加载方法返回的对象唯一id。 |
| 必须的 | 链接索引 | int | 链接索引或-1的基础。 |
| 必须的 | forceObj | vec3, 3个浮点数的列表 | 要施加的力/力矩矢量。请参见有关坐标系的标志。 |
| 必须的 | posObj | vec3, 3个浮点数的列表 | 在受力作用的连杆上的位置。仅适用于应用程序外部强制。请参见有关坐标系的标志。 |

| | | | |
|------|-----------------|-----|--|
| 必须的 | 旗 | int | 指定力/位置的坐标系：笛卡尔世界坐标为WORLD_FRAME，或局部链路坐标为LINK_FRAME。 |
| 可选择的 | physicsClientId | int | |

将返回物理服务器中出现的对象总数。

如果使用“getNumBodie”，则可以使用“getBodyuniqueid”查询主体唯一id。笔记所有api已经返回主体唯一id，所以如果您跟踪它们，通常不需要使用getBodyUniqueId。

getBodyInfo将返回从URDF、SDF、MJCF或其他文件中提取的基本名称和主体名称。

syncBodyInfo

当多个客户端连接到一个物理服务器改变世界（roadBodyIndfo）时，将同步身体信息（loadURDF，removyBody等）。

远程身体将通过其身体唯一的id（从loadURDF，loadSDF等）移除一个身体。

createConstraint, removeConstraint, changeConstraint

URDF、SDF和MJCF将铰接体指定为没有循环的树状结构。“创建约束”允许您连接特定的主体链接来关闭这些循环。参见子弹，子弹，四足动物。如何连接四足五杆闭环联动装置的腿。此外，您还可以在对象之间以及对象和特定世界框架之间创建任意约束。看

以Bullet/examples/pybullet/examples/constraint.py为例。

它还可以用来控制物理物体的运动，由动画帧驱动，如VR控制器。最好是使用约束条件，而不是直接设置位置或速度

为此目的，因为这些约束是与其他动力学约束一起解决的。创建约束条件具有以下输入参数：

| | | | |
|-----|--------------------|-----|---------------|
| 必须的 | parentBodyUniqueId | int | 父实体唯一ID |
| 必须的 | 父链接索引 | int | 父链接索引（或基数为-1） |

| | | | |
|------|-------------------|----------------|--|
| 必须的 | childBodyUniqueId | int | 子实体唯一ID，或-1表示无实体（在世界坐标中指定非动态子框架） |
| 必须的 | 子链接索引 | int | 子链接索引，或-1作为基础 |
| 必须的 | 接头类型 | int | 关节类型： JOINT_PRISMATIC、JOINT_FIXED、JOINT_POINT2POINT、JOINT_GEAR |
| 必须的 | 关节轴 | vec3, 3个浮点数的列表 | 关节轴，在子链接框架中 |
| 必须的 | 父框架位置 | vec3, 3个浮点数的列表 | 关节框架相对于质量框架的母心的位置。 |
| 必须的 | 子帧位置 | vec3, 3个浮点数的列表 | 关节框架相对于给定子质心框架的位置（如果未指定子框架，则为世界原点） |
| 可选择的 | 父框架方向 | vec4, 4个浮点数的列表 | 关节坐标系相对于母质心坐标系的方向 |
| 可选择的 | 子框架方向 | vec4, 4个浮点数的列表 | 关节框架相对于子重心坐标系的方向（如果没有指定子坐标系，则为世界原点框架） |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

创建约束将返回一个整数唯一id，可用于更改或删除约束。请看到

“examples/pybullet/examples/mimicJointConstraint.” JOINT_GEAR的例子和例子/小子弹/例子/微型龙。py为JOINT_POINT2POINT和示例/小符号/示例/约束。JOINT_FIXED的py。

更改约束

更改约束条件允许您更改现有约束条件的参数。输入参数包括：

| | | | |
|------|------------------------|----------------|--|
| 必须的 | userConstraintUniqueId | int | 由“创建约束”返回的唯一id |
| 可选择的 | 联合子枢轴 | vec3, 3个浮点数的列表 | 更新的子轴，请参见“创建约束” |
| 可选择的 | 联合子框架方向 | vec4, 4个浮点数的列表 | 更新的子框架方向为四个一组 |
| 可选择的 | maxForce | 使漂浮 | 约束条件可以施加的最大力 |
| 可选择的 | 齿轮比 | 使漂浮 | 两个齿轮旋转的速率之间的比率 |
| 可选择的 | gearAuxLink | int | 在某些情况下，如差速器驱动器，第三个（辅助）链路被用作参考姿态。 请参见 racecar differential。py |

| | | | |
|------|-----------------|-----|----------------------|
| 可选择的 | 相对位置目标 | 使漂浮 | 两个齿轮之间的相对位置目标偏移量 |
| 可选择的 | erp | 使漂浮 | 约束误差减少参数 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

请参见示例、示例/约束。py

远程约束将删除由其唯一id给出的约束。其输入参数为：

| | | | |
|------|------------------------|-----|---------------|
| 必须的 | userConstraintUniqueId | int | 由创建约束”返回的唯一id |
| 可选择的 | physicsClientId | int | 由“连接”返回的唯一id |

getNumConstraints, getConstraintUniqueId

您可以查询使用“创建约束”创建的约束的总数。可选参数是int物理客户端id。

getConstraintUniqueId

将取一个在0.范围内的串行索引，并报告约束的唯一id。请注意，约束的唯一id可能不是连续的，因为您可以删除约束。输入是整数串行索引，可选地是物理的sclientid。

getConstraintInfo/State

您可以查询约束信息，并给出一个约束唯一id。

输入参数为

| | | | |
|------|--------------------|-----|---------------|
| 必须的 | constraintUniqueId | int | 由创建约束”返回的唯一id |
| 可选择的 | physicsClientId | int | 由“连接”返回的唯一id |

输出列表为：

| | | |
|--------------------|-----|---------|
| parentBodyUniqueId | int | 请参见创建约束 |
| 母联索引 | int | 请参见创建约束 |

| | | |
|-------------------|----------------|---------|
| childBodyUniqueId | int | 请参见创建约束 |
| 子链接索引 | int | 请参见创建约束 |
| 约束类型 | int | 请参见创建约束 |
| 关节轴 | vec3, 3个浮点数的列表 | 请参见创建约束 |
| 父联轴器 | vec3, 3个浮点数的列表 | 请参见创建约束 |
| 儿童关节枢轴 | vec3, 3个浮点数的列表 | 请参见创建约束 |
| 联合框架方向父 | vec4, 4个浮点数的列表 | 请参见创建约束 |
| 联合框架方向儿童 | vec4, 4个浮点数的列表 | 请参见创建约束 |
| maxAppliedForce | 使漂浮 | 请参见创建约束 |
| 齿轮比 | 使漂浮 | 请参见创建约束 |
| gearAuxLink | int | 请参见创建约束 |
| 相对位置目标 | 使漂浮 | 请参见创建约束 |
| erp | 使漂浮 | 请参见创建约束 |

获取约束状态

给出一个约束唯一id，您可以在最近的模拟步骤中查询所应用的约束力。输入是一个约束唯一id，输出是一个约束力的向量，它的维数是受约束影响的自由度（例如，一个固定的约束影响6自由度）。

getDynamicsInfo/changeDynamics

你可以获得关于质量、质心、摩擦和基础和链接的其他属性的信息。

获取动态信息的输入参数为：

| | | | |
|------|-----------------|-----|----------------------|
| 必须的 | bodyUniqueId | int | 对象唯一id，由loadURDF等返回。 |
| 必须的 | 链接索引 | int | 链接（接头）索引或-1。 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

退货信息有限，我们需要时会提供更多信息：

| | | |
|-------------|----------------|---|
| 一团 | 两倍的 | 质量单位: kg |
| 横向摩擦 | 两倍的 | 摩擦系数 |
| 局部惯性 对角线 | vec3, 3个浮点数的列表 | 局部惯性对角线。请注意, 链接和底座以质心为中心, 并与惯性主轴对齐。 |
| 局部惯性位置 | vec3 | 惯性坐标系在关节坐标系局部坐标中的位置 |
| 局部惯性角 | vec4 | 关节系局部坐标系中惯性系的定位 |
| 归还原主 | 两倍的 | 回弹系数 |
| 滚动摩擦 | 两倍的 | 正交的滚动摩擦系数 |
| 旋转摩擦 | 两倍的 | 在接触法线附近的旋转摩擦系数 |
| 接触阻尼 | 两倍的 | -如果不可用, 则为1。接触约束的阻尼。 |
| 接触刚度 | 两倍的 | -如果不可用, 则为1。接触约束的刚度。 |
| 体型 | int | 1=刚体, 2=多体, 3=软体 |
| 碰撞边缘 | 两倍的 | 高级/内部/不受支持的信息。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。 |

更改动态

您可以使用变化动力学来更改质量、摩擦系数和恢复系数等特性。

输入参数为:

| | | | |
|------|--------------|-----|--|
| 必须的 | bodyUniqueId | int | 对象唯一id, 由loadURDF等返回。 |
| 必须的 | 链接索引 | int | 链接索引或-1的基础 |
| 可选择的 | 一团 | 两倍的 | 更改链接的质量 (或链接Index-1的基数) |
| 可选择的 | 横向摩擦 | 两倍的 | 横向 (线性) 接触摩擦 |
| 可选择的 | 旋转摩擦 | 两倍的 | 在接触法线周围的扭转摩擦力 |
| 可选择的 | 滚动摩擦 | 两倍的 | 扭转摩擦正交于接触法线 (保持这个值非常接近于零, 否则模拟可能会变得非常不现实)。 |
| 可选择的 | 归还原主 | 两倍的 | 接触的弹性。保持它略小于1, |

| | | | |
|------|----------------------|------|---|
| | | | 最好接近0。 |
| 可选择的 | linearDamping | 两倍的 | 链路的线性阻尼（默认为0.04） |
| 可选择的 | angularDamping | 两倍的 | 链接的角度阻尼（默认为0.04） |
| 可选择的 | 接触刚度 | 两倍的 | 触点约束的刚度，与触点阻尼一起使用。 |
| 可选择的 | contactDamping | 两倍的 | 此车身/链接的接触约束的阻尼。与接触性硬度一起使用。如果在联系人部分的URDF文件中指定了该值，则这将覆盖该值。 |
| 可选择的 | 摩擦力锚杆 | int | 启用或禁用摩擦锚：摩擦漂移校正（默认情况下禁用，除非在URDF接触部分中设置） |
| 可选择的 | localInertiaDiagnoal | vec3 | 惯性张量的对角线元素。请注意，基础和链接以质心为中心，并与惯性主轴对齐，因此在惯性张量中没有非对角线元素。 |
| 可选择的 | ccdSweptSphereRadius | 使漂浮 | 对球体的半径进行连续的碰撞检测。看 Bullet/examples/pybullet/examples/experimentalCcdS phereRadius. 例如py。 |
| 可选择的 | 联系人处理阈值 | 使漂浮 | 距离低于此阈值的接触点将由约束求解器进行处理。例如，如果接触处理阈值=为0，则距离为0.01的接触将不会作为约束进行处理。 |
| 可选择的 | 激活状态 | int | 当启用睡眠时，不移动（低于阈值）的对象将被禁用为睡眠，如果所有其他影响该阈值的对象也准备好了睡眠。 pybullet. 激活状态睡眠 pybullet. ACTIVATION_STATE_ENABLE_SLEEPING 炸药子弹。ACTIVATION_STATE_DISABLE_WAKEUP 您也可以使用标志= pybullet.URDF_ENABLE_SLEEPING 在loadURDF中启用睡眠。请参阅睡眠.py的例子。 |
| 可选择的 | jointDamping | 两倍的 | 在每个关节处应用的关节阻尼系数。该系数是从URDF关节阻尼场中读取出来的。将该值保持在接近0时。 关节阻尼力=-阻尼系数*关节速度。 |
| 可选择的 | anisotropicFriction | 两倍的 | 各向异性摩擦系数允许摩擦在不同方向的比例。 |
| 可选择的 | maxJointVelocity | 两倍的 | 给定关节的最大关节速度，如果在约束求解过程中超过，则会被夹紧，以避免破坏模拟。请确保您的模拟不超过最大的连接速度。 |

| | | | |
|------|-----------------|-----|---|
| | | | 默认的最大关节速度为100个单位。 |
| 可选择的 | 碰撞裕度 | 两倍的 | 无支持的更改碰撞裕度。根据形状类型，它可能向也可能不向碰撞形状的内部或外部添加一些填充。 |
| 可选择的 | 接头下限 | 两倍的 | 更改一个关节的下限，也需要关节的上限，否则它将被忽略。请注意，目前，关节限制并没有更新在“得到关节信息”！ |
| 可选择的 | 接头上限 | 两倍的 | 更改一个关节的上限，也要求关节下限，否则它将被忽略。请注意，目前，关节限制并没有更新在“得到关节信息”！ |
| 可选择的 | 联合极限力 | 两倍的 | 改变用于满足联合极限的最大力。 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

设置时间步骤

警告：在很多情况下，最好让时间步骤保持为默认值，即240Hz。我们已经根据这个值调整了几个参数。例如，求解器的迭代次数和接触、摩擦和非接触关节的误差减少参数（erp）与时间步长有关。如果您更改了时间步长，您可能需要相应地重新调整这些值，特别是erp值。

您可以设置在调用“步进模拟”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。不要定期改变这个时间步长。

设置时间步骤也可以使用新的设置物理工程参数API来实现。输入参数为：

| | | | |
|------|-----------------|-----|-----------------------------|
| 必须的 | 时间步 | 使漂浮 | 每次你调用“步骤模拟”时，时间步将继续进行“时间步”。 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

设置物理引擎参数

您可以使用设置物理工程参数API来设置物理引擎参数。此时，将显示以下输入参数：

| | | | |
|------|---------------------|-----|--|
| 可选择的 | 固定时间步骤 | 使漂浮 | 请参见设置时间步骤部分中的警告。物理引擎时间步长在几秒，每次你叫“步模拟”模拟时间将进展这个量。与“设置时间步骤”相同 |
| 可选择的 | numSolverIterations | int | 选择约束求解器迭代次数的最大次数。如果达到求解器剩余阈值，求解器可以在numSolverIterations. |
| 可选择的 | 使用分裂脉冲 | int | 先进特性，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，以防止巨大的渗透恢复力。 |
| 可选择的 | 分裂脉冲穿透阈值 | 使漂浮 | 与“使用分裂脉冲”相关：如果一个特定的接触约束的穿透小于这个指定的阈值，那么该接触将不会发生分裂脉冲。 |
| 可选择的 | numSubSteps | int | 用“数值子步骤”进一步细分物理模拟步骤。这将交换性能和准确性。 |
| 可选择的 | 碰撞过滤器模式 | int | 使用0作为默认的碰撞过滤器：（组A和maskB）和（groupB和maskA）。使用1切换到OR碰撞过滤器：（组A和maskB）OR（组B和maskA） |
| 可选择的 | 接触断开阈值 | 使漂浮 | LCP求解器不会处理距离超过此阈值的接触点。此外，aabb被这个数字扩展。在项目符号2. x中，默认值为0.02。 |
| 可选择的 | maxNumCmdPer1ms | int | 实验：如果执行的命令数量超过这个阈值，则添加1 ms睡眠。 |
| 可选择的 | enableFileCaching | int | 设置为0以禁用文件缓存，例如。obj波前文件加载 |
| 可选择的 | 恢复速度阈值 | 使漂浮 | 如果相对速度低于这个阈值，恢复量将为零。 |
| 可选择的 | erp | 使漂浮 | 约束误差减少参数（非接触、非摩擦） |
| 可选择的 | 联系ERP | 使漂浮 | 接触误差减少参数 |
| 可选择的 | 摩擦ERP | 使漂浮 | 摩擦误差降低参数（启用位置摩擦锚时） |
| 可选择的 | 启用锥形摩擦 | int | 设置为0以禁用隐式锥体摩擦和 |

| | | | |
|------|-------------------------|-----|---|
| | | | 使用金字塔近似值（圆锥体为默认值）。注意：虽然默认情况下启用，但如果存在摩擦因素，值得尝试禁用此特性。 |
| 可选择的 | 确定性重叠对 | int | 设置为1以启用，设置为0以禁用重叠对的排序（向后兼容性设置）。 |
| 可选择的 | allowedCcdPenetration | 使漂浮 | 如果启用了连续碰撞检测（CCD），则如果穿透率低于此阈值，则不使用CCD。 |
| 可选择的 | 联合反馈模式 | int | 快速联合反馈框架： JOINT_FEEDBACK_IN_WORLD_SPACE JOINT_FEEDBACK_IN_JOINT_FRAME |
| 可选择的 | solverResidualThreshold | 两倍的 | 速度阈值，如果每个约束的最大速度级误差低于此阈值，则求解器将终止（除非求解器命中求解次数）。默认值为1e-7。 |
| 可选择的 | contactSlop | 使漂浮 | 触点的位置校正不能通过低于此阈值进行解决，以允许更稳定的触点。 |
| 可选择的 | 启用SAT | int | 如果为true/1，则启用基于分离轴定理的凸碰撞检测，如果特征可用（而不是使用GJK和EPA）。在负载URDF中需要URDF_INITIALIZE_SAT_FEATURES。请参见“卫星碰撞”。py的例子。 |
| 可选择的 | constraintSolverType | int | 实验性的（最好忽略一下）：允许使用一个直接的LCP求解器，如Dantzig。看到switchConstraintSolverType.py的示例。 |
| 可选择的 | globalCFM | 使漂浮 | 实验（最好忽略）全局默认约束混合参数。 |
| 可选择的 | minimumSolverIslandSize | int | 实验（最好忽略），最小大小的约束解决岛屿，以避免非常小的岛屿的独立约束。 |
| 可选择的 | reportSolverAnalytics | int | 当true/1时，有额外的解分析。 |
| 可选择的 | 热启动因子 | 使漂浮 | 用于初始化初始求解器解的前一帧力/脉冲的分数 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

这是一个设置默认联系人参数设置的API。它将被卷入到设置的物理工程参数API中。

获取物理引擎参数

您可以使用可选的“获取物理引擎参数”命令和“物理sclientid”来查询一些当前的物理引擎参数。这将返回参数的命名元组。

重置模拟

重置模拟将从世界中删除所有对象，并将世界重置为初始条件。

| | | | |
|------|-----------------|-----|---|
| 可选择的 | 旗 | int | 实验性的标志，最好是忽略一下。 reset_use_simple_broadphase 重置可变形的世界 重置的_使用的_离散的世界 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

它接受一个可选的参数：物理客户端Id（以防您创建了多个物理服务器连接）。

startStateLogging/stopStateLogging

状态日志记录允许您记录模拟的状态，例如在每个模拟步骤之后的一个或多个对象的状态（在每次调用步骤模拟之后或在设置时间模拟后的每个模拟步骤之后的状态）。这将允许您记录对象的运动轨迹。还可以选择记录物体的共同状态，如基础位置和方向、关节位置（角度）和关节运动力。

所有使用开始状态日志记录生成的日志文件都可以使用C++或Python脚本读取。请参阅四重播放.py和kuka_with_cube_回放.py为Python脚本读取日志文件。你可以使用子弹3/示例。在C++中读取日志文件。

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，包括日志记录VR控制器的状态。

作为一个特殊案例，我们实现了微型机器人的记录。在PyBullet中生成的日志文件模拟结果与真实的微型龙四足动物日志文件相同。看Bullet/examples/pybullet/examples/logMinitaur.例如py。

重要提示：各种日志记录器都包括它们自己的内部时间戳，它在创建时从零开始。这意味着您需要同时启动所有的日志记录器，以保持同步。你需要做

当然，模拟不是在实时模式下，在启动记录器时运行：使用小符号。在创建日志记录器之前，请设置实际时间模拟(0)。

| | | | |
|------|-----------------|-------|--|
| 必须的 | 日志记录类型 | int | <p>实现了各种类型的日志记录。</p> <p>STATE_LOGGING_MINITAURO: 这将需要加载四足动物/四足动物。urdf和对象的唯一id。它记录时间戳、IMU滚动/俯仰/偏航、8个腿电机位置(q0-q7)、8个腿电机扭矩(u0-u7)、躯干的前进速度和模式(在模拟中未使用)。</p> <p>STATE_LOGGING_GENERIC_ROBOT: 这将记录所有对象或所选对象的数据日志(如果提供了对象统一对象的话)。</p> <p>STATE_LOGGING_VIDEO_MP4: 这将打开一个MP4文件，并开始使用ffmpeg管道将OpenGL 3D可视化处理器像素流化到该文件中。它将需要安装ffmpeg。您也可以使用avconv(在Ubuntu上的默认值)，只需创建一个符号链接，这样ffmpeg就可以指向avconv。在Windows上，ffmpeg在某些情况下会导致撕裂/颜色伪影。</p> <p>state_logint_联系点</p> <p>state_logging_vr_controllers .</p> <p>状态日志记录配置文件时间 这将转储一个计时文件，可以使用谷歌Chrome打开：//跟踪负载。</p> |
| 必须的 | 文件名 | 细绳 | 存储日志文件数据的路径名(绝对或相对路径)。 |
| 可选择的 | objectUniqueIds | int列表 | 如果保留为空，日志记录器可以记录每个对象，否则日志记录器只记录对象uniqueid列表中的对象。 |
| 可选择的 | maxLogDof | int | 要记录的关节自由度的最大数目(不包括基本数量)。这也适用于STATE_LOGGING_GENERIC_ROBOT_DATA。默认值为12。如果一个机器人超过了dofs的数量，它就根本不会被记录下来。 |
| 可选择的 | bodyUniqueIdA | int | 适用于STATE_LOGGING_CONTACT_POINTS。如果提供，则只记录涉及车身统一性的接触点。 |
| 可选择的 | bodyUniqueIdB | int | 适用于STATE_LOGGING_CONTACT_POINTS。如果提供，则只记录涉及车身单位指数的接触点。 |
| 可选择的 | 链接索引A | int | 适用于STATE_LOGGING_CONTACT_POINTS。如果提供，只日志接触点涉及linkInqueIdA。 |
| 可选择的 | linkIndexB | int | 适用于STATE_LOGGING_CONTACT_POINTS。如果提供，只日志接触点涉及链接的身体unikueida。 |
| 可选择的 | 设备类型过滤器 | int | <p>设备类型过滤器允许您选择什么VR设备记录： VR_DEVICE_CONTROLLER, VR_DEVICE_HMD , VR_DEVICE_GENERIC_TRACKER或它们的任何组合。</p> |

| | | | |
|------|-----------------|-----|--|
| | | | 适用于STATE_LOGGING_VR_CONTROLLERS。默认值为VR_DEVICE_CONTROLLER。 |
| 可选择的 | 日志标志 | int | （即将到来的PyBullet 1.3.1）。STATE_LOG_JOINT_TORQUES，以记录由关节电机引起的关节扭矩。 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

该命令将返回一个非负的int记录uniqueid，可以与停止状态记录一起使用。

任务事项：记录为每种日志记录类型记录的数据。现在，使用日志读取实用程序来查找，或检查C++ [根源 编码 的 那 日志记录或Python dumpLog.py脚本。](#)

停止状态日志记录

您可以使用它的日志uniqueid停止一个记录器。

提交配置文件时间

提交配置文件时间允许插入开始和停止时间来配置Python代码。[请参阅概要时间.py](#)的例子。

小子弹和子弹已经安装了许多功能，所以你可以看到时间花在哪里。您可以将这些配置文件时间转储到一个文件中，它可以使用谷歌Chrome在关于：[//使用加载功能的跟踪窗口中查看](#)。在GUI中，您可以按“p”来启动/停止配置文件转储。在某些情况下，您可能需要仪器化客户机代码的计时。您可以使用PyBullet提交配置文件计时。下面是一个输出示例：

变形和布（FEM，PBD）

除了铰接多体和刚体动力学，小子弹还实现了变形和布模拟，使用有限元方法（FEM）质量弹簧系统以及使用基于位置的动力学（PBD）。一些使用该实现的研究论文是[学习 重新排列范围 可变形的 电缆， 织物， 和 包 和 目标调节的 运输网络 （使用有限元法，ICRA2021），Sim-to-Real 加强 学习 为了 可变形的 对象操作 （使用PBD）和辅助功能 健身房 A 物理学 仿真 框架 为了 辅助机器人。](#)（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（请参见下面创建的软人体锚）。

默认情况下，PyBullet将使用基于位置的动力学（PBD）。您可以启用基于有限元方法（FEM）的模拟：

```
pybullet.resetSimulation (p. 重置使用变形的世界)
```

[参见可变形的环面。](#)以Py为例。

loadSoftBody/loadURDF

有几种方法可以创建可变形的对象，包括布料或体积对象。
加载软体允许您从VTK或OBJ文件中加载一个可变形的对象。

以下参数可用于加载软主体。请注意，有几个参数假设您使用了有限元模型，并且对PBD模拟没有影响。

| | | | |
|------|------------------------|------|-------------------------------------|
| 必须的 | 文件名 | 细绳 | 可变形的文件名。可以在波前面。obj格式或VTK格式。 |
| 可选择的 | 基本位置 | vec3 | 可变形物体的初始位置 |
| 可选择的 | 基底方向 | vec4 | 可变形对象的初始取向（四元数x、y、z、w） |
| 可选择的 | 规模 | 两倍的 | 调整可变形对象大小的缩放因子（默认为= 1） |
| 可选择的 | 一团 | 两倍的 | 可变形的总质量，质量均匀分布在所有顶点之间 |
| 可选择的 | 碰撞裕度 | 两倍的 | 碰撞边缘扩展了可变形的，它可以帮助避免穿透，特别是对于薄（布）可变形的 |
| 可选择的 | 使用质量弹簧 | 布尔 | 使用质量弹簧 |
| 可选择的 | 使用弯曲弹簧 | 布尔 | 创建弯曲弹簧来控制的弯曲可变形的 |
| 可选择的 | 使用NeoHooke | 布尔 | 启用新胡克式模拟 |
| 可选择的 | 弹簧弹性刚度 | 两倍的 | 刚度参数 |
| 可选择的 | springDampingStiffness | 两倍的 | 阻尼参数 |

| | | | |
|------|--------------------|-----|----------------------|
| 可选择的 | 弹簧阻尼所有方向 | 两倍的 | 弹簧阻尼参数 |
| 可选择的 | 弹簧弯曲刚度 | 两倍的 | 弯曲刚度参数 |
| 可选择的 | 新胡克安姆 | 两倍的 | 新胡克恩模型的参数 |
| 可选择的 | NeoHooke和Lambda | 两倍的 | 新胡克恩模型的参数 |
| 可选择的 | NeoHooke破坏 | 两倍的 | 新胡克恩模型的参数 |
| 可选择的 | frictionCoeff | 两倍的 | 可变形材料的接触摩擦 |
| 可选择的 | 使用面部接触 | 布尔 | 启用与面内部的碰撞，而不仅仅是在顶点上。 |
| 可选择的 | 使用自我碰撞 | 布尔 | 使可变形的自碰撞 |
| 可选择的 | repulsionStiffness | 两倍的 | 一个可以帮助避免渗透的参数。 |

loadURDF

PyBullet还扩展了URDF格式，以使用“可变形”标签指定可变形对象。[例如，请参见可变形的环面。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>
    <碰撞_保证金value= "0.006" />
    <斥力_刚度值= "800.0" />
    <friction value= "0.5"/>
    <neohookean mu= "60" lambda= "200" damping= "0.01" />
    <visual filename="torus.vtk"/>
  </deformable>
</robot>
```

创建软主体锚

您可以将可变形对象的顶点固定到世界上，或使用创建软体锚器将可变形对象的顶点附加到多体上。这将返回一个约束的唯一id。您可以使用远程约束API删除此约束。

[请参见可变形的锚。](#)例如py。

您可以使用getMeshData API访问可变形对象的顶点。

合成相机渲染

PyBullet有一个内置的OpenGL GPU可视化器和一个基于Tiny渲染器的内置的CPU渲染器。这使得从任意的相机位置渲染图像非常容易。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，例如在谷歌云平台或Colab中的云渲染。[请看电子邮件渲染测试](#)。例如如何使用它，如“插件”部分所述。

合成相机由两个4比4的矩阵指定：视图矩阵和投影矩阵。由于这些不是很直观，有一些辅助方法来从可理解的参数计算视图和投影矩阵。

看看这个[文章](#)关于内在的相机矩阵与链接到OpenGL相机信息。

computeView/ProjectionMatrix

计算视矩阵输入参数为

| | | | |
|------|-----------------|----------------|--------------------|
| 必须的 | 相机的眼睛位置 | vec3, 3个浮点数的列表 | 眼位坐标 在笛卡尔世界 |
| 必须的 | 相机目标位置 | vec3, 3个浮点数的列表 | 目标（焦点）点的位置 |
| 必须的 | 相机向上向量 | vec3, 3个浮点数的列表 | 照相机的向上矢量，在笛卡尔世界坐标中 |
| 可选择的 | physicsClientId | int | 未使用，为API一致性添加 |

输出是4x4的视图矩阵，存储为包含16个浮点数的列表。

computeViewMatrixFromYawPitchRoll

输入参数为

| | | | |
|-----|--------|--------|----------------|
| 必须的 | 相机目标位置 | 3个浮子列表 | 笛卡尔世界坐标系中的目标焦点 |
| 必须的 | 距离 | 使漂浮 | 从眼睛到焦点的距离 |
| 必须的 | yaw | 使漂浮 | 围绕上轴的左右偏角。 |
| 必须的 | 场地 | 使漂浮 | 以上下的程度俯仰。 |
| 必须的 | 卷 | 使漂浮 | 围绕向前向量滚动 |

| | | | |
|------|-----------------|-----|-----------------|
| 必须的 | 上轴指数 | int | Y轴向上是1，Z轴向上是2。 |
| 可选择的 | physicsClientId | int | 未使用的，添加的API一致性。 |

输出是4x4的视图矩阵，存储为包含16个浮点数的列表。

计算投影矩阵

输入参数为

| | | | |
|------|-----------------|-----|-----------------|
| 必须的 | 左边的 | 使漂浮 | 左屏幕（画布）坐标 |
| 必须的 | 右边的 | 使漂浮 | 右屏幕（画布）坐标 |
| 必须的 | 底部的 | 使漂浮 | 底部屏幕（画布）坐标 |
| 必须的 | 最高的 | 使漂浮 | 顶部屏幕（画布）坐标 |
| 必须的 | 靠近 | 使漂浮 | 近平面距离 |
| 必须的 | 遥远地 | 使漂浮 | 远平面距离 |
| 可选择的 | physicsClientId | int | 未使用的，添加的API一致性。 |

输出是4x4投影矩阵，存储为16个浮点数的列表。

computeProjectionMatrixFOV

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

输入参数为：

| | | | |
|------|-----------------|-----|-----------------|
| 必须的 | fov | 使漂浮 | 视场 |
| 必须的 | 方面 | 使漂浮 | 纵横比 |
| 必须的 | Val附近 | 使漂浮 | 近平面距离 |
| 必须的 | farVal | 使漂浮 | 远平面距离 |
| 可选择的 | physicsClientId | int | 未使用的，添加的API一致性。 |

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割掩模缓冲区，每个像素的可见对象的主体唯一id。注意，PyBullet可以使用numpy选项编译：使用numpy将提高复制相机的性能

从C到Python的像素。注意：旧的渲染图像API已经过时，被camage取代。

getCamera图像输入参数：

| | | | |
|------|--------------------|----------------|--|
| 必须的 | 宽度 | int | 像素单位的水平图像分辨率 |
| 必须的 | 高度 | int | 垂直图像的分辨率，单位为像素 |
| 可选择的 | 查看矩阵 | 16个浮子 | 4x4个视图矩阵，请参见计算机视图矩阵* |
| 可选择的 | 投影矩阵 | 16个浮子 | 4x4投影矩阵，请参见计算机投影* |
| 可选择的 | 光的方向 | vec3, 3个浮点数的列表 | 灯光方向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。 |
| 可选择的 | 浅色 | vec3, 3个浮点数的列表 | 0范围内的红色，绿色，蓝色。1，仅适用于ER_TINY_RENDERER |
| 可选择的 | 光距离 | 使漂浮 | 光沿归一化光方向的距离，仅适用于ER_TINY_RENDERER |
| 可选择的 | 影子 | int | 1表示阴影，0表示无阴影，仅适用于ER_TINY_RENDERER |
| 可选择的 | lightAmbientCoeff | 使漂浮 | 光环境系数，仅适用于er_tiny_renderer |
| 可选择的 | lightDiffuseCoeff | 使漂浮 | 光漫射系数，仅适用于er_tiny_renderer |
| 可选择的 | lightSpecularCoeff | 使漂浮 | 光镜面系数，仅适用于er_tiny_renderer |
| 可选择的 | 渲染器 | int | ER_BULLET_HARDWARE_OPENGL或ER_TINY_RENDERER。请注意，直接模式没有OpenGL，所以它需要ER_TINY_RENDERER。 |
| 可选择的 | 旗 | int | ER_SEGMENTATION_MASK_OBJECT_AND_LINKINDEX，见下面的描述 分割面具缓冲区和示例代码。使用ER_NO_SEGMENTATION_MASK来避免计算分割掩码。 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

返回一个参数列表：

| | | |
|------------------------|-----------------------------------|---|
| 宽度 | int | 宽度图像分辨率在（水平 像素）中 |
| 高度 | int | 高度图像分辨率（垂直） 像素 |
| rgbPixels | 列表[红色，绿色，蓝色，字母]][0..width*高度] | R、G、B、A格式的像素颜色列表， 范围[0..255] |
| 深度像素 | 浮动列表[0..width*height] | 深度缓存弹头使用OpenGL进行渲染， 并且约定是 非线性z缓冲区。看 https://stackoverflow.com/questions/6652253/ 从m深度缓冲区获得真正的z值 投影矩阵上的far=1000. //depends ，这是投影矩阵上的默认 near=0.01//depends 深度=远很近（很远（很近）是子弹的深度 另请参见PyBullet https://github.com/bulletphysics/bullet3/blob/master/examples/pybullet/examples/pointCloudFromCameraImage.py |
| segmentationMaskBuffer | int [0..的列表。width*height] | 对于每个像素，可见对象的唯一id。如果 使用ER_SEGMENTATION_MASK_OBJECT_AND_LINKINDEX，分割面具缓冲区将对象唯一id和链接索引如下所示： 值=对象uniqueId+（链接索引+1） <<24。 请参见示例。 因此，对于一个没有关节/链接的自由浮动体，分割掩码等于它的主体的唯一id，因为它的链接索引是-1。 。 |

isNumpyEnabled

请注意，对于大型图像，从C/C++复制像素到Python可能非常慢，除非您使用NumPy编译PyBullet。您可以检查是否启用NumPy

PyBullet.isNumpyEnabled(). pip安装项目已安装了NumPy。目前，只有使用numpy加速。

获取可视化形状数据

您可以使用get可视化形状数据访问可视化形状信息。您可以使用它来连接您自己的渲染方法与PyBullet模拟，并在每个模拟步骤之后手动同步世界转换。您还可以使用getMeshData，特别是对于可变形的对象，来接收关于顶点位置的数据。

输入参数为：

| | | | |
|----------|-----------------|-----|---|
| 必须的 | objectUniqueId | int | 对象唯一id，由加载方法返回。 |
| optionsl | 旗 | int | VISUAL_SHAPE_DATA_TEXTURE_UNIQUE_IDS也将提供textureUniqueId |
| 可选择的 | physicsClientId | int | 物理客户端id为由“连接”返回 |

输出是视觉形状数据列表，每个视觉形状的格式如下：

| | | |
|-----------------|----------------|---|
| objectUniqueId | int | 对象唯一id，与输入相同 |
| 链接索引 | int | 链接索引或-1的基础 |
| 视觉几何图形类型 | int | 视觉几何图形类型（TBD） |
| 按规格尺寸切割 | vec3, 3个浮点数的列表 | 几何图形的尺寸（尺寸、局部比例） |
| 网格资产文件名 | 字符串，字符列表 | 到三角形网格的路径，如果有的话。通常相对于URDF、SDF或MJCF文件位置，但也可以是绝对的。 |
| 本地可视化帧位置 | vec3, 3个浮点数的列表 | 局部视觉框架的位置，相对于连接/连接框架的位置 |
| 本地可视化框架方向 | vec4, 4个浮点数的列表 | 局部视觉框架相对于链接/关节框架的方向 |
| rgbaColor | vec4, 4个浮点数的列表 | URDF颜色（如果有指定的）为红色/绿色/蓝色/alpha |
| textureUniqueId | int | (字段仅存在于在使用VISUAL_SHAPE_DATA_TEXTURE_UNIQUE_IDS标志) 纹理唯一的id的形状，或-1，如果没有 |

物理模拟使用质心作为笛卡尔世界变换的参考，在得到基础位置和方向和在得到链接状态。如果实现了自己的渲染，则需要将局部视觉变换转换为世界空间，利用质心世界变换和（逆）局部惯性框架。您可以使用getLink状态API访问本地信息框架。

changeVisualShape, loadTexture

您可以使用更改可视化形状来更改形状的纹理、RGBA颜色和其他属性。

| | | | |
|------|-----------------|----------------|---|
| 必须的 | objectUniqueId | int | 对象唯一id，由加载方法返回。 |
| 必须的 | 链接索引 | int | 环比指数 |
| 可选择的 | 形状指数 | int | 实验为内部使用，建议忽略形状指数或留下它-1。目的是让你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一个视觉形状。这个形状索引与getVisual形状数据返回的列表排序相匹配。 |
| 可选择的 | textureUniqueId | int | 纹理唯一id，由“加载纹理”方法返回 |
| 可选择的 | rgbaColor | vec4，列表4 彩车 | 红色、绿色、蓝色和ALPHA的颜色组件，每个都在范围内[0..1]。Alpha当前必须是0（不可见）或1（可见）。请注意，修补渲染器不支持透明，但GUI/EGL OpenGL3渲染器支持。 |
| 可选择的 | specularColor | vec3 | 镜面颜色成分，红色，绿色和蓝色，可以从0到大的数量（>100）。 |
| 必须的 | physicsClientId | int | 物理客户端id为由“连接”返回 |

负载纹理

从文件加载纹理，如果加载成功，返回非负纹理唯一id。这个唯一的id可以与更改的可视化形状一起使用。

碰撞检测查询

您可以查询在上一个“步骤模拟”或“执行碰撞检测”中存在的接触点信息。要获得接触点，您可以使用“获取接触点”API。请注意，“获取接触点”将不会重新计算任何接触点信息。

getOverlappingObjects, getAABB

此查询将返回所有具有轴对齐边界框与给定轴对齐边界框重叠的对象的唯一id。注意，查询是保守的，可能返回没有实际AABB重叠的其他对象。这是因为加速度

结构有一些启发式的，可以扩大aabb（额外的边缘和沿速度矢量挤压）。

对象输入参数为：

| | | | |
|------|-----------------|----------------|----------------------|
| 必须的 | aabbMin | vec3, 3个浮点数的列表 | aabb的最小坐标 |
| 必须的 | aabbMax | vec3, 3个浮点数的列表 | aabb的最大坐标 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

获取重叠对象将返回一个对象唯一id的列表。

getAABB

给定对象唯一id和链接索引，可以查询轴对齐边界框（在世界空间中）。（当您不传递链接索引或使用-1时，您将得到基的AABB）。

输入参数为

| | | | |
|------|-----------------|-----|--------------------------------|
| 必须的 | bodyUniqueId | int | 由创建方法返回的对象唯一id。 |
| 可选择的 | 链接索引 | int | 链接索引在范围内[0..getNumJoints (..)] |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

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

[参见getAABB。](#)py的例子。

getContactPoints, getClosestPoints

获取接触点API返回在最近一次对逐步模拟或执行碰撞检测的调用期间计算出的接触点。请注意，如果在逐步模拟之后更改模拟的状态或执行碰撞检测，则“获取接触点”不会更新，并且可能无效。其输入参数如下：

| | | | |
|------|-------|-----|--------------|
| 可选择的 | bodyA | int | 只报告涉及身体A的接触点 |
|------|-------|-----|--------------|

| | | | |
|------|-----------------|-----|--|
| 可选择的 | bodyB | int | 只报告涉及身体B的接触点。重要提示：如果你提供身体B，你需要有一个有效的身体A。 |
| 可选择的 | 链接索引A | int | 只报告涉及人体A的链接indexA的接触点 |
| 可选择的 | linkIndexB | int | 仅报告涉及车身连接的接触点 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

获取接触点将返回一个接触点的列表。每个接触点都有以下字段：

| | | |
|---------------------|----------------|--------------------|
| 联系人标志 | int | 预订的 |
| bodyUniqueIdA | int | 身体A的身体唯一id |
| bodyUniqueIdB | int | 身体B的身体唯一id |
| 链接索引A | int | 主体A的链接索引，-1为基数 |
| linkIndexB | int | 主体B的链接指数，-1为基数 |
| positionOnA | vec3, 3个浮点数的列表 | 在笛卡尔世界坐标 |
| positionOnB | vec3, 3个浮点数的列表 | 在B上的接触位置，在笛卡尔世界坐标中 |
| contactNormalOnB | vec3, 3个浮点数的列表 | 接触B上的法线，指向A |
| 接触距离 | 使漂浮 | 接触距离，分离为正，穿透为负 |
| 法向力 | 使漂浮 | 在最后一次“逐步模拟”中施加的法向力 |
| lateralFriction1 | 使漂浮 | 横向摩擦力1方向上的横向摩擦力 |
| lateralFrictionDir1 | vec3, 3个浮点数的列表 | 第一横向摩擦方向 |
| lateralFriction2 | 使漂浮 | 横向摩擦力2方向上的横向摩擦力 |
| lateralFrictionDir2 | vec3, 3个浮点数的列表 | 第二个横向摩擦方向 |

获得最接近的点

它也可以计算最近的点，独立于逐步模拟或执行碰撞检测。这还允许您以任意的分离距离计算对象的最近点。在此查询中，将不会报告任何正常的力。

获取关闭点数的输入参数：

| | | | |
|-----|-------|-----|-----------------|
| 必须的 | bodyA | int | 第一个对象(A)的对象唯一id |
|-----|-------|-----|-----------------|

| | | | |
|------|-----------------|-----|-----------------------------|
| 必须的 | bodyB | int | 第二个对象(B)的对象唯一id |
| 必须的 | 距离 | 使漂浮 | 如果对象之间的距离超过此最大距离，则可能不返回任何点。 |
| 可选择的 | 链接索引A | int | 对象A的链接索引（基数为-1） |
| 可选择的 | linkIndexB | int | 对象B的链接索引（基准值为-1） |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

返回与接触点相同格式的最近点列表（但在这种情况下，正常力总是为零）

rayTest, rayTestBatch

您可以执行一次光线投射，以找到第一个命中对象的交集信息。

光线测试的输入参数为：

| | | | |
|------|-----------------|----------------|----------------------|
| 必须的 | 射线从位置 | vec3, 3个浮点数的列表 | 在世界坐标中，光线的开始 |
| 必须的 | 射线到位置 | vec3, 3个浮点数的列表 | 在世界坐标中的射线的末端 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

如果出现交叉点，光线测试查询将返回以下信息：

| | | |
|----------------|----------------|--------------------------|
| objectUniqueId | int | 对象被命中对象的唯一id |
| 链接索引 | int | 命中对象的链接索引，如果没有/父对象，则为-1。 |
| 命中分数 | 使漂浮 | 沿射线的[0, 1]范围内。 |
| 命中位置 | vec3, 3个浮点数的列表 | 在笛卡尔人的世界坐标系中的命中位置 |
| 达到正常水平 | vec3, 3个浮点数的列表 | 在笛卡尔世界坐标中达到正常值 |

射线测试批次

这类似于rayTest，但允许您提供一个射线阵列，以便更快地执行。“光线到位置”的大小需要等于“光线到位置”的大小。即使没有交集，您也可以得到一个光线结果：您需要使用对象UniqueId字段来检查光线是否命中了任何东西：如果对象UniqueId是-1，则没有命中。在这种情况下，“命中分数”是1。每批处理的最大光线数为
pybullet.max_ray_intersection_batch_size .

光线测试程序批处理输入参数为：

| | | | |
|------|----------------------|------------------|--|
| 必须的 | 射线来自位置 | vec3的列表，3个浮点数的列表 | 世界坐标中每条光线的起点列表 |
| 必须的 | 射线到位置 | vec3的列表，3个浮点数的列表 | 世界坐标中每条射线的端点列表 |
| 可选择的 | parentObjectUniqueId | int | 来自/的光线位于父对象的局部空间中 |
| 可选择的 | 父链接索引 | int | 来自/的光线位于父对象的局部空间中 |
| 可选择的 | numThreads | int | 使用多个线程来计算射线测试（0 =使用所有可用的线程，正数=正好是这个线程的量，默认的=-1 =单线程） |
| 可选择的 | 报告命中次数 | int | 你可以报告第n次命中，而不是第n次命中 |
| 可选择的 | 碰撞过滤器掩码 | int | 只有当碰撞过滤器掩码和身体碰撞过滤器组之间的位和非零时，测试命中。有关如何修改车身过滤器遮罩/组，请参阅设置碰撞过滤器组遮罩。 |
| 可选择的 | 部分Epsilon | 使漂浮 | 只有在使用报告时有用：当命中相同的体时，如果分数与现有的命中相似，则忽略重复命中。例如，一条光线可能会击中一个物体的许多共面三角形，你可能只对其中一个击中物感兴趣。 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

输出是每个输入光线的一个光线交叉结果，其信息与上面的rayTest查询相同。请参见batchRayTest.py示例，如何使用它。

获取碰撞形状数据

您可以使用此查询查询现有主体基础和链接的碰撞几何体类型和其他碰撞形状信息。它的工作原理与获取可视化的形状数据非常相似。

获取碰撞的形状数据的输入参数为：

| | | | |
|-----|----------------|-----|---------------------|
| 必须的 | objectUniqueId | int | 对象唯一id，从loadURDF等接收 |
| 必须的 | 链接索引 | int | 链接索引或-1的基础 |

| | | | |
|------|-----------------|-----|----------------------|
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |
|------|-----------------|-----|----------------------|

返回值是一个包含以下内容的列表：

| | | |
|---------|------|--|
| 对象唯一ID | int | 对象唯一ID |
| 链接索引 | int | 链接索引或-1的基础 |
| 几何形状类型 | int | 几何图形类型： GEOM_BOX、GEOM_SPHERE、GEOM_CAPSULE、GEOM_MESH、GEOM_PLANE |
| 按规格尺寸切割 | 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，维度是比例因子。 |
| 文件名 | 细绳 | 仅针对GEOM_MESH： 碰撞网格资源的文件名（和路径） |
| 局部帧位置 | vec3 | 碰撞坐标系相对于质心/惯性坐标系的局部位置。 |
| 局部框角 | vec4 | 碰撞坐标系相对于惯性坐标系的局部方向。 |

启用/禁用碰撞

默认情况下，会在不同的动态运动体之间启用碰撞检测。可以使用loadURDF中的“URDF_USE_SELF_COLLISION”标志来启用同一主体的链接之间的自碰撞（有关更多信息，请参阅loadURDF命令）。

您可以使用设置的碰撞过滤器组掩码API来启用和禁用对象组之间的碰撞检测。

V-HACD

小子弹包括体积层次近似分解（vhacd）的实现，由Khaled Mamou。这可以导入一个凹形的波前面。obj文件，并导出一个包含凸分解部分的新的波前obj文件。这可以用于小子弹有效地处理凹运动几何。

对于静态（非移动）凹三角形网格环境，您可以使用URDF文件中的标记（<链接凹=“是”名称=“>）或使用标记=将三角形网格标记为凹。geom_force_concave_trimesh .

| | | | |
|------|-------------------------|-----|--|
| 必须的 | 文件名 | 细绳 | 源（凹面）波前obj文件名 |
| 必须的 | 文件名称输出 | 细绳 | 目标（凸分解）波前obj文件名 |
| 必须的 | 文件名日志 | 细绳 | 日志文件名 |
| 可选择的 | 凹面 | 两倍的 | 最大允许凹度（default=0.0025、range=0.0-1.0） |
| 可选择的 | 阿尔法 | 两倍的 | 控制对沿着对称平面进行剪切的偏差（default=0.05，range=0.0-1.0） |
| 可选择的 | 贝塔 | 两倍的 | 控制对沿着旋转轴进行裁剪的偏差（default=0.05，range=0.0-1.0） |
| 可选择的 | 伽马 | 两倍的 | 控制合并阶段中允许的最大凹度（default=0.00125，range=0.0-1.0） |
| 可选择的 | minVolumePerCH | 两倍的 | 控制生成的凸壳的自适应采样（default=0.0001，range=0.0-0.01） |
| 可选择的 | 决心 | int | 在体素化阶段中生成的最大体素数（默认=100,000，范围=10,000-16,000,000） |
| 可选择的 | maxNumVerticesPerCH | int | 控制每个凸包的最大三角形数量（默认=64，范围=4-1024） |
| 可选择的 | 深 | int | 最大的剪裁阶段数。在每个分割阶段，根据最佳切割平面（默认=20，范围=1-32）剪切=凹度高于用户定义阈值的部分 |
| 可选择的 | planeDownsampling | int | 控制搜索“最佳”剪切平面的粒度（默认为=4，范围为=1-16） |
| 可选择的 | 凸型船体结构放大器 | int | 控制在剪切平面选择阶段的凸壳生成过程的精度（默认=4，范围=1-16） |
| 可选择的 | pca | int | 在应用凸分解之前启用/禁用网格归一化（默认=0，范围={0,1}） |
| 可选择的 | 方式 | int | 0：基于体素的近似凸分解，1：基于四面体的近似凸分解（默认为=0，范围为={0,1}） |
| 可选择的 | convexhullApproximation | int | 在计算凸包时启用/禁用近似值（默认=1，范围={0,1}） |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。注意：vhacd分解目前发生在客户端。 |

示例用法：

导入piture作为p

导入pybullet_data作为pd

导入操作系统

```
p.connect (p. 直接)
name_in = os. 路径。加入 (pd.getDataPath ( ) , “duck.obj” )
name_out =” duck_vhacd2. obj”
name_log =” 日志.txt”
p.vhacd (name_in, name_out, name_log)
```

设置碰撞过滤器组掩码

每个身体都是一个群体的一部分。如果他们的组与面具匹配，它会与其他身体发生碰撞，反之亦然。以下检查是使用有关两具尸体的小组和面罩进行的。这取决于碰撞过滤器的模式。

| | | | |
|------|-----------------|-----|----------------------|
| 必须的 | bodyUniqueId | int | 要配置的车身单位 |
| 必须的 | 链接索引A | int | 要配置的主体的链接索引 |
| 必须的 | 碰撞过滤器组 | int | 过滤器的按位排列的组，请参见下面的解释 |
| 必须的 | 碰撞过滤器掩码 | int | 过滤器的按位掩码，见下面的解释 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

您可以对特定链接对之间的碰撞检测进行更精细的控制。在这里，使用集合碰撞过滤器对API：您可以启用或禁用碰撞检测。设置碰撞过滤器对将覆盖过滤器组/掩模和其他逻辑。

设置碰撞过滤器对

| | | | |
|-----|---------------|-----|--------------------|
| 必须的 | bodyUniqueIdA | int | 要过滤的主体A的身体单位 |
| 必须的 | bodyUniqueIdB | int | 身体B将被过滤，A==B意味着自碰撞 |
| 必须的 | 链接索引A | int | 体A的链接索引 |
| 必须的 | linkIndexB | int | B体的连接指数 |
| 必须的 | 启用碰撞 | int | 1启用碰撞，0禁用碰撞 |

| | | | |
|------|-----------------|-----|----------------------|
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |
|------|-----------------|-----|----------------------|

[还有一个插件API来编写您自己的碰撞过滤实现，请参见碰撞过滤器插件 实施.](#)

逆动力学，运动学

calculateInverseDynamics (2)

计算反向动力学将从指定的关节位置和速度开始，计算达到给定的关节加速度所需的力。逆动力学的计算使用递归牛顿欧拉算法（RNEA）。

计算的平均动态输入参数为：

| | | | |
|------|------------------|------|---|
| 必须的 | bodyUniqueId | int | 主体唯一id，由负载URDF返回等。 |
| 必须的 | objPositions | 浮子列表 | 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）。 |
| 必须的 | objVelocities | 浮子列表 | 每个自由度的联合速度（DoF） |
| 必须的 | objAccelerations | 浮子列表 | 每个自由度的期望关节加速度（DoF） |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

计算反向动力学返回每个自由度的联合力的列表。
请注意，当涉及多个（球形）关节时，计算的平均动态值会使用不同的代码路径，而且速度有点慢。还要注意的，计算反向动力学忽略了关节/链接阻尼，而正向动力学（在逐步模拟中）包括这些阻尼项。因此，如果你想比较逆动力学和正向动力学，确保使用连接阻尼和连接阻尼设置这些阻尼项为零，通过线性阻尼和角度阻尼连接阻尼。

calculateJacobian, MassMatrix

雅可比矩阵将计算链接上的一个点的平移和旋转雅各比矩阵。 $g.x_dot = J * q_dot$. 返回的雅各比亚人略有不同，这取决于是否

根链接是固定的或浮动的。如果是浮动的，雅各比亚语将包括对应于根链接自由度的列；如果是固定的，雅各比亚语将只有与关节相关联的列。函数调用需要运动状态的完整描述，这是因为计算逆动力学实际上是首先调用的，从中提取期望的雅各比矩阵；因此，如果需要的话，通过联合速度和加速度的零矢量是合理的。

计算得到的雅可比矩阵输入参数为：

| | | | |
|------|------------------|------|----------------------------------|
| 必须的 | bodyUniqueId | int | 主体唯一id，由负载URDF返回等。 |
| 必须的 | 链接索引 | int | 为雅可比亚语的链接索引。 |
| 必须的 | 本地位置 | 浮子列表 | 指定链接上的计算雅可比矩阵的点，在其质心周围的局部坐标的链接中。 |
| 必须的 | objPositions | 浮子列表 | 接头位置（角度） |
| 必须的 | objVelocities | 浮子列表 | 联合速度 |
| 必须的 | objAccelerations | 浮子列表 | 期望的关节加速度 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

计算雅可比矩阵返回：

| | | | |
|-----|-----------------|-----------------------------------|------------------------------------|
| 必须的 | linearJacobian | mat3x ((dof), (dof), (dof)) | 翻译雅可比语， $x_dot = J_t * q_dot$ 。 |
| 必须的 | angularJacobian | mat3x ((dof), (dof), (dof)) | 旋转雅可比琴， $r_dot = J_r * q_dot$ 。 |

计算质量矩阵

计算质量矩阵将计算给定关节位置的系统惯性。采用复合刚体算法（CBRA）计算质量矩阵。

| | | | |
|------|-----------------|------------|----------------------|
| 必须的 | bodyUniqueId | int | 主体唯一id，由负载URDF返回等。 |
| 必须的 | objPositions | 阵列 的 浮动 | 每个链接的连接位置。 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

结果是维数为dofCount的平方质量矩阵，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，当涉及多个（球形）关节时，计算的质量矩阵将使用不同的代码路径，这要慢一些。

反运动学

您可以计算使末端执行器在笛卡尔世界空间中到达给定目标位置的关节角度。在内部，子弹使用了一个改进版本的塞缪尔巴斯逆运动学库。目前，只有有或没有零空间控制的阻尼最小二乘方法被暴露出来，并且有一个末端执行器目标。您还可以选择指定末端执行器的目标方向。此外，还有一个选项可以使用空空间来指定关节限制和休息姿态。此可选的空空间支持需要所有4个列表（低限值、上限、连接范围、恢复位置），否则将使用常规IK。参见反运动学。示例/示例文件夹中的py示例。

计算反运动学(2)

计算逆运动学输入参数为：

| | | | |
|------|--------------|----------------|--|
| 必须的 | bodyUniqueId | int | 主体唯一id，由loadURDF返回 |
| 必须的 | 端部效应链路索引 | int | 末端执行器链路索引 |
| 必须的 | 目标位置 | vec3, 3个浮点数的列表 | 末端执行器的目标位置（其链接坐标，而不是质心坐标！）。默认情况下，这是在笛卡尔世界空间，除非您提供当前位置关节角度。 |
| 可选择的 | 目标方向 | vec3, 4个浮点数的列表 | 笛卡尔世界空间中的目标取向，四元数[x, y, w, z]。如果没有指定，则将使用纯位置IK。 |
| 可选择的 | 下限 | 浮动列表[0.. nDof] | 可选的空空间IK需要所有4个列表（低限制、上限限制、连接范围、重新位置）。否则，将使用常规的IK。仅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，下限、上限、连接范围很容易导致IK解决方案中的冲突和不稳定。首先尝试使用一个广泛的范围和限制，只有剩下的姿势。 |
| 可选择的 | 上限 | 浮动列表[0.. nDof] | 可选的空空间IK需要所有4个列表（低限制、上限限制、连接范围、重新位置）。.. 否则，常规IK将使用下限限值和上限限值指定联合限制 |
| 可选择的 | 联合范围 | 浮动列表[0.. nDof] | 可选的空空间IK需要所有4个列表（低限制、上限限制、连接范围、重新位置）。否则，将使用常规的IK。 |
| 可选择的 | 休息姿势 | 浮动列表[0.. nDof] | 可选的空空间IK需要所有4个列表（低限制、上限限制、连接范围、重新位置）。否则，将使用常规的IK.. |

| | | | |
|------|------------------|----------------|--|
| | | | 选择一个更接近一个给定的休息姿势的IK解决方案 |
| 可选择的 | jointDamping | 浮动列表[0.. nDof] | 关节阻尼允许使用关节阻尼系数来调整IK解决方案 |
| 可选择的 | 解决者 | 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“逆运动学的选择性阻尼最小二乘”所述。 |
| 可选择的 | 当前位置 | 浮动列表[0.. nDof] | 联合位置列表。默认情况下，“子弹”使用主体的关节位置。如果提供，目标位置和目标定位是在本地空间！ |
| 可选择的 | 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，或达到最大的数值迭代值。默认为20次迭代。 |
| 可选择的 | 剩余阈值 | 两倍的 | 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，或达到最大的数值迭代值。 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

计算逆运动学返回每个自由度的关节位置列表，因此此列表的长度是关节的自由度数（跳过基础关节和固定关节）。例子请参见Bullet/examples/pybullet/inverse_kinematics.py。

默认情况下，IK将改进解决方案，直到目标末端执行器和实际末端执行器之间的距离低于剩余阈值（ $1e-4$ ）或达到最大迭代次数。

计算反运动学2

类似于计算逆运动学，但它需要一系列末端执行器指数及其目标位置（目前没有方向）。

| | | | |
|-----|--------------|--------|---|
| 必须的 | bodyUniqueId | int | 主体唯一id，由loadURDF返回 |
| 必须的 | 端部效应器链接索引 | int列表 | 末端执行器链路索引 |
| 必须的 | 目标位置 | vec3列表 | 末端执行器的目标位置（其链接坐标，而不是质心坐标！）。默认情况下，这是在笛卡尔世界空间中， |

| | | | |
|-----|----------------------|--|----------------|
| | | | 除非你提供当前位置关节角度。 |
| ... | 有关其他参数，请参见 计算反运动学 | | |

强化学习健身房

在“pip安装项目符号”期间安装了一套RL体育馆环境。这包括OpenAI健身房环境的子弹版本，如蚂蚁、跳跃斗、人形和步行者。也有一些环境适用于模拟和真实的机器人，如幽灵机器人微型龙四足动物，麻省理工学院赛车和KUKA机器人手臂抓取环境。

小子弹、pybullet_envs、pybullet_data的源代码和示例如下：

<https://github.com/bulletphysics/bullet3/tree/master/examples/pybullet/gym>.

您可以使用诸如DQN、PPO、TRPO和DDPG等RL训练算法来训练环境。有几个预先训练过的例子，你可以像这样享受它们：

安装小子弹，张量流，健身房

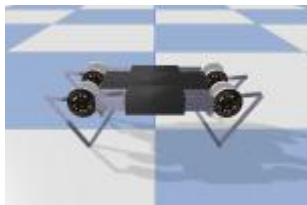
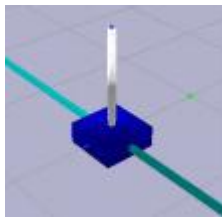






```
巨蟒-m pybullet_envs。例子enjoy_TF_HumanoidBulletEnv_v0_2017may蟒蛇-m  
pybullet_envs.examples.kukaGymEnvTest
```


环境和数据

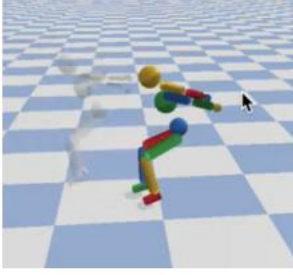




在“sudopip安装小子弹”之后，pybullet_envs和pybullet_data包可用。导入pybullet_envs包将自动注册环境到OpenAI健身房。

你可以使用以下Python行获得健身房的子弹环境列表：

打印






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

| 环境名称 | 描述 |
|---|---|
| MinitaurBulletEnv-v0  | <p>幽灵机器人在平坦地面上模拟四足动物。该环境用于RSS 2018年的“模拟到真实：四足机器人的学习敏捷运动”，见论文在...上 arxiv.</p> <p>基于旅行距离的奖励。使用健身房： env =健身房创建课程。</p> <pre>make('MinitaurBulletEnv-v0')</pre> <p>或者使用该类直接创建环境，并使用以下参数：</p> <p>导入pybullet_envs。子弹minitaur_gym_env作为e</p> <pre>env = e.MinitaurBulletEnv (render=True)</pre> |
| 人类模仿背部脂肪公告环境 -v1和人类模仿行走 BulletEnv-v1 | <p>在小子弹中深度模拟论文的重新实现：模拟参考运动。该实现允许选择参考运动。后空翻和行走的参考运动都变成了一个健身房的环境。</p> <p>各种运动的预训练模型可作为小子弹的一部分：需要张力流1. x (1.14)：</p> <pre>python3 -m pybullet_envs.深模拟。特斯特尔——arg_file run_humanoid3d_backflip_args .txt</pre> |

| | |
|---|--|
|  | <pre>python3 -m pybullet_envs.深模拟。特斯特尔——arg_file run_humanoid3d_walk_args .txt</pre> <p>您还可以使用所包含的DeepMimic MPI并行化PPO实现进行训练：</p> <pre>蟒蛇3 mpi_run。皮——arg_file train_humanoid3d_walk_args. txt——num_workers 16</pre> <p>（根据您的机器更改内核的数量）要回放您训练的策略，您需要复制权重。</p> |
| <p>请参见视频， 以及一个使用Human 3.6 数据集的例子。</p> | |
| <p>RacecarBulletEnv-v0</p>  | <p>麻省理工学院遥控赛车的仿真。根据与随机放置的球之间的距离进行奖励。观察结果是照相机帧中的球形位置（x、y）。环境的作用空间可以是离散的（对于DQN），也可以是连续的（对于PPO、TRPO和DDPG）。</p> <p>导入pybullet_envs。子弹赛车GymEnv</p> <pre>env = e.RacecarGymEnv (isDiscrete=False ,renders=True)</pre> <p>env. 重置</p> |
| <p>RacecarZedBulletEnv-v0</p>  | <p>和赛车公告env-v0一样，但观察结果是相机的像素。</p> |
| <p>KukaBulletEnv-v0</p>  | <p>模拟KUKA Liwa机械臂，抓住托盘中的物体。主要的奖励发生在最后，当抓住一定高度的物体。每一步都会发生一些非常小的奖励/成本：行动的成本和握把和物体之间的距离。观测包括物体的x，y位置。</p> <p>注意：这个环境目前有培训的问题，我们来调查一下。</p> |
| <p>KukaCamBulletEnv-v0</p>  | <p>和库卡子弹战一样，但观察是相机的像素。</p> |

[我们移植了机器人学校 环境pybuner](#)。机器人学校的环境比MuJoCo体育馆的环境更难。

| | |
|-----------------|-------|
| AntBulletEnv-v0 | 蚂蚁更重， |
|-----------------|-------|

| | |
|--|--------------------------------|
|  | 鼓励它通常在地面上有两条或更多的腿。 |
| HalfCheetahBulletEnv-v0  | |
| HumanoidBulletEnv-v0  | 类人猿受益于更现实的能量成本（=扭矩×角速度）从奖励中减去。 |
| HopperBulletEnv-v0  | |
| Walker2DBulletEnv-v0  | |
| InvertedPendulumBulletEnv-v0 | |
| InvertedDoublePendulumBulletEnv-v0 | |
| InvertedPendulumSwingupBulletEnv-v0 | |

它也可以访问的数据，如URDF/SDF机器人资产，波前。来自pybullet_data中的OBJ文件包。
下面是一个此操作示例：

```

导入喷枪
导入pybullet_data
数据路径= pybullet_data.getDataPath()
枪弹。连接(枪弹。图形用户界面
pybullet.setAdditionalSearchPath (datapath)
pybullet.loadURDF("r2d2.urdf",[0,0,1])

```

或者，在loadURDF/SDF命令中手动将数据路径附加到文件名中。

稳定的基线和ARS, ES, ……。

对于连续控制健身房环境，如半猎豹（v0），蚂蚁（AntBulletEnv_v0），（AntBulletEnv_v0），Hopper）HopperBulletEnv_v0，连续子弹-v0，你可以使用稳定 [基线](#) 下面是一个示例：

```

pip3安装稳定的基线-用户
pip3安装pybuner—用户
python3 -m pybullet_envs.稳定的基线。火车，火车，火车，一半，子弹

```

要享受训练过的环境，请将权重文件复制/重命名为
sac_HalfCheetahBulletEnv-v0.zip（删除_best部分）

```

python3 -m pybullet_envs.稳定的基线。享受，享受，享受，享受，享受，享受，享受，享受，
享受，第五集

```

[稳定的 基线](#) 动物园提供预先训练过的小子弹环境。

您也可以在谷歌Colab笔记本中使用稳定基线来训练和享受小子弹环境，请参见这个Colab [样例](#)
[的训练 a cartpole.](#)

训练和享受： DQN, PPO, ES

对于离散的健身环境，如库卡公告v-v0和比赛公告v-v0，你可以使用OpenAI [基线DQN使用一个离散的动作空间来训练模型](#)。本文提供了一些例子，即如何训练和享受这些离散的环境：

```

巨蟒-m pybullet_envs.基线train_pybullet_cartpole
巨蟒-m pybullet_envs.基线train_pybullet_racecar

```

OpenAI基线将保存一个。当模型改进时，以指定的时间间隔提交PKL文件。这个PKL文件在享受脚本中使用：

```
巨蟒-m pybullet_envs。基线enjoy_pybullet_cartpole
巨蟒-m pybullet_envs。基线enjoy_pybullet_racecar
```

PyBullet还附带一些预先训练过的模型，你可以享受开箱即用。以下是一些预先训练过的环境的列表：

```
巨蟒-m pybullet_envs。例子enjoy_TF_AntBulletEnv_v0_2017may
巨蟒-m pybullet_envs。例子enjoy_TF_HalfCheetahBulletEnv_v0_2017may
巨蟒-m pybullet_envs。例子enjoy_TF_AntBulletEnv_v0_2017may
巨蟒-m pybullet_envs。例子enjoy_TF_HopperBulletEnv_v0_2017may
巨蟒-m pybullet_envs。例子enjoy_TF_HumanoidBulletEnv_v0_2017may
巨蟒-m pybullet_envs。例子enjoy_TF_InvertedDoublePendulumBulletEnv_v0_2017may蟒蛇-m
pybullet_envs。例子enjoy_TF_InvertedPendulumBulletEnv_v0_2017may蟒蛇-m pybullet_envs
。例子enjoy_TF_InvertedPendulumSwingupBulletEnv_v0_2017may蟒蛇-m pybullet_envs。例子
enjoy_TF_Walker2DBulletEnv_v0_2017may
```

使用张量流和脉冲火炬的列车

您可以使用张量流代理训练各种子弹环境 [仅限于事前许可](#) 首先安装

所需的Python包： pip安装健身房，紧张流，代理，小子弹，宝石。然后用于培训：

```
dir
```

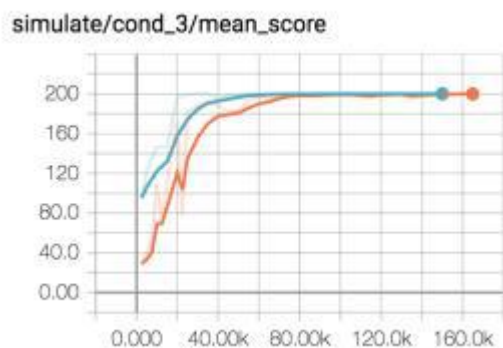
以下环境可作为代理配置使用：

```
pybullet_pendulum
pybullet_doublependulum
pybullet_pendulumswingup
pybullet_cheetah
pybullet_ant
pybullet_racecar
pybullet_minitaur
```

您可以使用拉伸板查看培训进展：

```
tensorboard --logdir=pendulum --port=2222
```

打开一个web浏览器并访问本地主机：2222页。这里是一个来自张力板的钟摆训练的例子图：



训练结束后，您可以可视化训练后的模型，创建一个视频或使用物理服务器 (pybullet_envs pybullet_envs。例子在物理服务器模式或虚拟现实)。如果启动本地GUI物理服务器，可视化工具 (meton_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色调渲染器来代替。若要生成视频，请使用：

巨蟒-m pybullet_envs。代理人visualize_ppo——logdir=摆/xxxxx——超越=摆_视频，以类似

的方式，你可以训练和可视化微型龙机器人：

巨蟒-m pybullet_envs。代理人train_ppo——配置=pybullet_minitaaur——
logdir=pybullet_minitaaur这里是一个微型龙步态的例子视频：<https://www.youtube.com/watch?v=tfqCHDoFHRQ>

进化策略（ES）

在<http://brog>上有一篇由DavidHa（硬丸人）撰写的博客文章，即如何使用进化策略训练小球子弹环境。[奥托罗。2017年11月12日/发展稳定战略](#)

.

使用PyTorch PPO列车

我们将添加一些描述如何开始使用PyTorch和小子弹。与此同时，请查看这个存储库：
<https://github.com/ikostrikov/pytorch-a2c-ppo-acktr>

虚拟现实

另请参见[vrBullet 快速启动 指南](#)

VR物理服务器使用OpenVR API为HTC Vive和Oculus Rift触摸控制器支持。OpenVR当前阀门也在一个Linux上工作 [变体](#)

参见<https://www.youtube.com/watch?v=VMJyZtHQL50>

一个例子，子弹的一部分，可以完全控制使用小子弹在共享内存，UDP或TCP关系



对于Windows上的VR，建议编译Bullet物理SDK使用微软Visual Studio (MSVC)。生成MSVC项目文件，通过运行“build_visual_studio_vr_pybullet_double”。蝙蝠”脚本。您可以自定义这个小脚本来指向Python的位置等。确保切换到MSVC的“释放”配置，并构建和运行App_PhysicsServer_SharedMemory_VR*.exe。默认情况下，这个VR应用程序将呈现一个显示跟踪器/控制器（如果可用的话）。

getVREvents, setVRCameraState

getVREvents将返回选定VR设备的列表事件，该设备自上次调用getVREvents以来状态改变了。当不提供任何设备类型筛选器时，默认情况是仅报告VR_DEVICE_CONTROLLER状态。你可以选择任何设备的组合，包括VR_DEVICE_CONTROLLER、VR_DEVICE_HMD（头戴设备）和VR_DEVICE_GENERIC_TRACKER（如HTC Vive跟踪器）。

请注意，VR_DEVICE_HMD和VR_DEVICE_GENERIC_TRACKER只报告位置和方向事件。getVREvents具有以下参数：

| | | | |
|------|-----------------|-----|---|
| 可选择的 | 设备类型过滤器 | int | 默认值为VR_DEVICE_CONTROLLER。您也可以选择VR_DEVICE_HMD或VR_DEVICE_GENERIC_TRACKER或它们的任何组合。 |
| 可选择的 | 所有模拟轴 | int | 1表示所有的模拟轴，0只有一个轴 |
| 可选择的 | physicsClientId | int | 如果您连接到多个服务器，您可以选择一台。 |

输出参数为：

| | | |
|-------------------|--------------------------------|--|
| controllerId | int | 控制器索引 (0..max_vr_controllers) |
| 控制器位置 | vec3, 3个浮点数的列表 | 控制器位置，在世界空间笛卡尔坐标中 |
| 控制器方向 | vec4, 4个浮点数的列表 | 世界空间中的控制器方向四元数[xyzw] |
| 控制器模拟轴 | 使漂浮 | 模拟轴值 |
| numButtonEvents | int | 自上次调用getVREvents以来的按钮事件数 |
| numMoveEvents | int | 自上次调用getVREvents以来的移动事件数 |
| 的复数 | int[64]，按钮状态列表（OpenVR最多有64个按钮） | 每个按钮的标志：VR_BUTTON_IS_DOWN（目前按住），VR_BUTTON_WAS_TRIGGERED（自上次到getVREvents至少下降了一次，VR_BUTTON_WAS_RELEASED（自上一次调用getVREvents至少发布了一次）。请注意，只有VR_BUTTON_IS_DOWN才会报告实际的当前状态。例如，如果按钮上下上升，你可以从释放/触发标志中分辨出来，即使IS_DOWN仍然是假的。注意，在日志文件中，这些按钮包含10个1个整数的按钮（每个按钮3位）。 |
| 设备类型 | int | 设备类型：VR_DEVICE_CONTROLLER，虚拟现实设备HMD或虚拟现实设备通用跟踪器 |
| 所有类似程序（仅当有明确要求时！） | 10个浮子列表 | 目前，MAX_VR_ANALOGUE_AXIS是5，对于每个轴x和y的值。 |

请参见/示例、项目/示例/vrEvents。VR绘图和子弹/示例/小子弹/示例/vrTracker。负责跟踪HMD和通用跟踪器。

setVRCameraState

设置虚拟相机状态允许设置相机根变换偏移位置和方向。这允许控制虚拟现实相机在虚拟世界中的位置。它也可以让VR摄像头跟踪一个物体，比如车辆。

setVRCameraState具有以下参数（没有返回值）：

| | | | |
|------|-----------------|---------------|--|
| 可选择的 | 根位置 | vec3, 3个浮动的向量 | 相机根位置 |
| 可选择的 | 根方向 | vec4, 4个浮动的向量 | 四元数格式的照相机根方向。 |
| 可选择的 | 轨道对象 | vec3, 3个浮动的向量 | 要跟踪的对象的唯一iD |
| 可选择的 | 跟踪对象标志 | int | 旗VR_CAMERA_TRACK_OBJECT_ORIENTATIO N（如果启用，则会同时跟踪位置和方向） |
| 可选择的 | physicsClientId | int | 如果您已连接到多个服务器，则您可以选择 |

| | | | |
|--|--|--|---|
| | | | 一 |
|--|--|--|---|

调试GUI、行、文本、参数

PyBullet有一些功能，可以简化调试、可视化和调整模拟过程。这个特性只有在有一些3D可视化窗口，例如GUI模式或连接到一个单独的物理服务器（例如“物理服务器”模式下的示例浏览器或带有OpenGLGUI的独立物理服务器）时才有用。

添加用户调试程序，行，文本，参数

您可以添加由三维起点（从）和终点（到）指定的三维线，一种颜色 [红，绿，蓝]，一条线宽和一个持续时间，以秒为单位。添加用户调试程序的参数是：

| | | | |
|------|----------------------|----------------|--|
| 必须的 | lineFromXYZ | vec3, 3个浮点数的列表 | 笛卡尔世界坐标线的起点 |
| 必须的 | lineToXYZ | vec3, 3个浮点数的列表 | 在笛卡尔世界坐标中的线的终点 |
| 可选择的 | lineColorRGB | vec3, 3个浮点数的列表 | RGB颜色 [红色、绿色、蓝色] 每个组件在范围 [0..1] |
| 可选择的 | 线宽 | 使漂浮 | 线宽（受OpenGL实现的限制） |
| 可选择的 | 生命周期 | 使漂浮 | 使用0作为永久行，或以秒为正时间（之后自动删除带的行） |
| 可选择的 | parentObjectUniqueId | int | 即将发布的PyBullet 1.0.8中的新版本：在父对象/链接的局部坐标中绘制线。 |
| 可选择的 | 父链接索引 | int | 即将发布的PyBullet 1.0.8中的新版本：在父对象/链接的局部坐标中绘制线。 |
| 可选择的 | replaceItemUniqueId | int | 替换现有行（提高性能并避免删除/添加闪烁）请参见 f10_racecar.py 的例子。 |
| 可选择的 | physicsClientId | int | 如果您已连接到多个服务器，则您可以选择一个服务器 |

它将返回一个非负的唯一id，允许您使用remee删除该行。（当使用“替换项”时，它将返回替换的ItemUniqueId）。

addUserDebugText

您可以使用颜色和大小在特定的位置添加一些三维文本。输入参数为：

| | | | |
|------|----------------------|----------------|--|
| 必须的 | 文本 | 文本 | 以字符串（字符数组）表示的文本 |
| 必须的 | 文本位置 | vec3，列表3 彩车 | 文本在笛卡尔世界坐标中的3d位置[x, y, z] |
| 可选择的 | textColorRGB | vec3，列表3 彩车 | RGB颜色[红色、绿色、蓝色]每个组件在范围[0..1] |
| 可选择的 | 文本大小 | 使漂浮 | 文本大小 |
| 可选择的 | 生命周期 | 使漂浮 | 使用0作为永久文本，或以秒为单位的正时间（之后将自动删除带有的文本） |
| 可选择的 | 文本方向 | vec4，列表4 彩车 | 默认情况下，调试文本将始终面对摄像机，并自动旋转。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（指定父对象时）。请注意，相机面向文本使用了不同的实现/着色器，具有不同的外观：相机面向文本使用位图字体，具有指定方向的文本使用TrueType字体.. |
| 可选择的 | parentObjectUniqueId | int | 即将发布的PyBullet 1.0.8中的新版本：在父对象/链接的局部坐标中绘制线。 |
| 可选择的 | 父链接索引 | int | 即将发布的PyBullet 1.0.8中的新版本：在父对象/链接的局部坐标中绘制线。 |
| 可选择的 | replaceItemUniqueId | int | 替换现有的文本项（以避免删除/添加的闪烁） |
| 可选择的 | physicsClientId | int | 如果您已连接到多个服务器，则您可以选择一个服务器 |

该文本将返回一个非负的唯一id，允许您使用remeve用户debugItem删除该行。请参阅项目/例子/DdawItems.py

addUserDebugParameter

添加用户调试参数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。这允许您使用读数用户调试参数来读取参数的值。addUser调试参数的输入参数为：

| | | | |
|-----|--------|-----|---------------------------|
| 必须的 | 票面金额名称 | 细绳 | 参数的名称 |
| 必须的 | 最小范围 | 使漂浮 | 最小值如果最小值>最大值，将出现一个按钮而不是滑块 |
| 必须的 | 最大范围 | 使漂浮 | 最大值 |

| | | | |
|------|-----------------|-----|--------------------------|
| 必须的 | 起始值 | 使漂浮 | 初值 |
| 可选择的 | physicsClientId | int | 如果您已连接到多个服务器，则您可以选择一个服务器 |

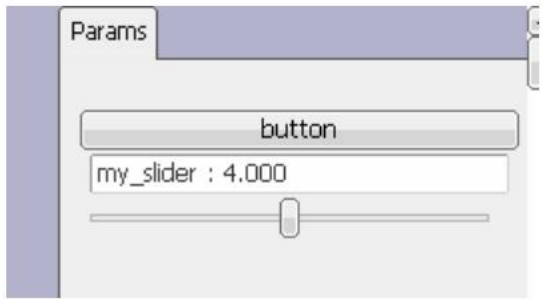
读取用户调试参数的输入参数为：

| | | | |
|------|-----------------|-----|-----------------------------|
| 必须的 | itemUniqueId | int | 由 ‘addUserDebug参数’)返回的唯一id |
| 可选择的 | physicsClientId | int | 如果您已连接到多个服务器，则您可以选择一个服务器 |

返回值是滑块中参数的最新读取值。对于一个按钮，每按下一个按钮，按钮的参数就会增加1。

样例

```
p. addUserDebugParameter("button", 1, 0, 1)
p. 添加用户调试参数 ( “我的滑块” , 3、5、 4)
```



删除所有用户参数

这将删除所有的滑块和按钮。

| | | | |
|------|-----------------|-----|--------------------------|
| 可选择的 | physicsClientId | int | 如果您已连接到多个服务器，则您可以选择一个服务器 |
|------|-----------------|-----|--------------------------|

removeUserDebugItem/All

添加用户调试行的函数，如果文本成功，将返回一个非负的唯一id。您可以使用远程用户调试项方法，使用此唯一id删除调试项。输入参数为：

| | | | |
|------|-----------------|-----|--------------------------|
| 必须的 | itemUniqueId | int | 要删除的调试项的唯一id（行、文本等） |
| 可选择的 | physicsClientId | int | 如果您已连接到多个服务器，则您可以选择一个服务器 |

removeAllUserDebugItems

此API将删除所有调试项（文本、行等）。

setDebugObjectColor

内置的OpenGL可视化处理器有一个线框调试渲染特性：按“w”进行切换。线框有一些默认的颜色。您可以覆盖一个特定对象的颜色，并链接使用设置调试对象颜色。输入参数为：

| | | | |
|------|---------------------|----------------|----------------------------------|
| 必须的 | objectUniqueId | int | 对象的唯一id |
| 必须的 | 链接索引 | int | 环比指数 |
| 可选择的 | objectDebugColorRGB | vec3, 3个浮点数的列表 | 调试颜色在[红色，绿色，蓝色]。如果不提供，自定义颜色将被删除。 |
| 可选择的 | physicsClientId | int | 如果您已连接到多个服务器，则您可以选择一个服务器 |

添加用户数据

简而言之，添加、删除和查询当前附加到主体的任何链接的文本字符串。请参见[userData.py的例子，关于如何使用它](#)。它将返回一个用户dDataId。请注意，您还可以在urdf文件中添加用户数据。

获取用户数据

根据附加用户数据返回的用户数据，获取将接收用户数据。请参见用户数据。例如使用py。

同步用户数据

同步用户数据将在多个客户端更改用户数据时同步用户数据（getUser数据等）。

删除用户数据

远程用户数据将删除以前添加的用户数据，给定一个用户的用户数据。

获取用户数据Id并获取NumUser数据

获取数字数据将返回给定一个主体单位的用户数据条目的数量。

获取用户数据信息

将用户数据的密钥和标识符检索为（用户数据DataId，键，身体统一id，链接索引，可视化形状索引）

configureDebugVisualizer

您可以配置内置的OpenGL可视化处理器的一些设置，例如启用或禁用线框、阴影和GUI渲染。这是有用的，因为一些笔记本电脑或桌面gui与我们的OpenGL 3可视化器有性能问题。

| | | | |
|------|--------------------|------|---|
| 必须的 | 旗 | int | 要启用或禁用的功能，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是默认的世界轴），COV_ENABLE_RGB_BUFFER_PREVIEW，COV_ENABLE_DEPTH_BUFFER_PREVIEW，COV_ENABLE_SEGMENTATION_MARK_PREVIEW |
| 必须的 | 使能够 | int | 0或1 |
| 可选择的 | 灯光位置 | vec3 | 可视化器灯的位置 |
| 可选择的 | 阴影MapRe解决方案 | int | 阴影贴图纹理的大小，对于许多gpu的功率通常为2。默认值为4096。现代gpu可以处理16384或32768或更高版本。 |
| 可选择的 | shadowMapWorldSize | int | 世界空间中阴影映射的大小（单位，默认为10） |
| 可选择的 | shadowMapIntensity | 使漂浮 | 阴影的可见性，0=不可见，1=完全可见 |
| 可选择的 | rgbBackground | vec3 | 背景颜色[红、绿、蓝 |
| 可选择的 | physicsClientId | int | 如果您已连接到多个服务器，则您可以选择一个服务器 |

样例

```
pybullet.configureDebugVisualizer (pybullet.cov_enable_wireframe ,1)
```

get/resetDebugVisualizerCamera

警告：获取虚拟照相机的返回参数的顺序与resetDebugVisualizerCamera. 不同将在未来的API版本（主要的新版本）中得到修复。

resetDebugVisualizerCamera

您可以重置3D OpenGL调试可视化器相机距离（在眼睛和相机目标位置之间）、相机偏航和间距和相机目标位置。

| | | | |
|------|-----------------|----------------|--------------------------|
| 必须的 | 摄影机距离 | 使漂浮 | 从眼睛到相机的目标位置的距离 |
| 必须的 | cameraYaw | 使漂浮 | 相机左右偏航角角度（度） |
| 必须的 | 摄影机间距 | 使漂浮 | 摄像机上下倾斜角度（单位是度） |
| 必须的 | 相机目标位置 | vec3, 3个浮点数的列表 | 相机目标位置是相机的焦点 |
| 可选择的 | physicsClientId | int | 如果您已连接到多个服务器，则您可以选择一个服务器 |

例如： `pybullet.resetDebugVisualizerCamera（相机距离=3，=30，=52，摄影师位置=[0,0,0]）`

getDebugVisualizerCamera

使用此命令可以获取相机的宽度和高度（像素）、视图和投影矩阵。输入参数是可选的物理客户端id。输出信息为：

| | | |
|-------|------------------|-------------------|
| 宽度 | int | 照相机图像的宽度，单位为像素 |
| 高度 | int | 照相机图像的像素的高度 |
| 查看矩阵 | 浮点数16，列表16 彩车 | 照相机的查看矩阵 |
| 投影矩阵 | 浮点数16，列表16 彩车 | 摄像机的投影矩阵 |
| 照相机向上 | 浮点3，列表3 彩车 | 相机的上轴，在笛卡尔世界空间坐标中 |
| 摄像机前进 | 浮点3，列表3 彩车 | 相机的前轴，在笛卡尔世界空间坐标中 |

| | | |
|-----|---------------|--|
| 水平的 | 浮点3，列表3 彩车 | TBD。这是一个可用来生成光线的水平矢量（例如用于鼠标选择或创建简单的光线跟踪器） |
| 垂直的 | 浮点3，列表3 彩车 | TBD。这是一个垂直向量，可以用于生成光线（例如，用于鼠标挑选或创建一个简单的光线跟踪器）。 |
| yaw | 使漂浮 | 照相机的偏航角，在笛卡尔局部空间坐标中 |
| 场地 | 使漂浮 | 照相机的俯仰角，在笛卡尔局部空间坐标中 |
| 分发 | 使漂浮 | 照相机和相机目标之间的距离 |
| 目标 | 浮点3，列表3 彩车 | 照相机的目标，在笛卡尔世界空间坐标中 |

getKeyboardEvents, getMouseEvents

您可以接收自上次调用“获取键盘事件”以来发生的所有键盘事件。每个事件都有一个密钥代码和一个状态。该状态是KEY_IS_DOWN、KEY_WAS_TRIGGERED和KEY_WAS_RELEASED的位标志组合。如果一个密钥从“上”到“下”状态，您将收到KEY_IS_DOWN状态以及KEY_WAS_TRIGGERED状态。如果一个键被按下并释放，状态将是键被下，键被释放。

一些特殊的键被定义为： 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_SHIFT, B3G_CONTROL, B3G_RETURN.

获取键盘事件的输入是一个可选的物理客户端i:

| | | | |
|------|---------------------|-----|--------------------------|
| 可选择的 | physicsClientI d | int | 如果您已连接到多个服务器，则您可以选择一个服务器 |
|------|---------------------|-----|--------------------------|

输出是一个键码“key”和键盘状态“值”的字典。
例如

```
qKey = ord('q')
键=p.get键盘事件 ( )
如果键和键。KEY_WAS_TRIGGERED: 打破;
```

获取鼠标事件

类似于获取键盘事件，您可以获得自上次调用获取鼠标事件以来发生的鼠标事件。所有的鼠标移动事件都被合并到一个具有最新位置的单个鼠标移动事件中。此外，一个给定按钮的所有鼠标按钮事件都将被合并。如果一个按钮上下上升，状态将是“KEY_WAS_TRIGGERED”。我们重用KEY_WAS_TRIGGERED /KEY_IS_DOWN /KEY_WAS_RELEASED。

获取鼠标事件的输入参数是：

| | | | |
|------|-----------------|-----|--------------------------|
| 可选择的 | physicsClientId | int | 如果您已连接到多个服务器，则您可以选择一个服务器 |
|------|-----------------|-----|--------------------------|

输出是以下格式的鼠标事件列表：

| | | |
|-----------|-----|--|
| 事件类型 | int | 鼠标_move_事件=1，鼠标_按钮_事件=2 |
| mousePosX | 使漂浮 | 鼠标指针的x坐标 |
| mousePosY | 使漂浮 | 鼠标指针的y坐标 |
| 按钮索引 | int | 鼠标左、中、右键的按钮索引 |
| 按钮状态 | 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/添加颜色的对象。](#)

插件

PyBullet允许您用C或C++编写插件来添加定制特性。PyBullet的一些核心特性是作为插件编写的，如PD控制、渲染、gRPC服务器、碰撞过滤和虚拟现实同步。大多数作为PyBullet核心部分的插件默认都是静态链接的，所以您不需要手动加载和卸载它们。

在Linux上，egl插件是一个默认附带PyBullet的插件的例子。它可以启用使用硬件OpenGL 3。没有X11上下文的X渲染，例如在谷歌云平台上的云渲染。[请参阅eglender测试。](#)例如，如何使用它。

PyBullet还附带了一个fileIO插件，它可以直接从zip文件加载文件，并允许文件缓存。[请参阅文件IOPlugin。](#)例如，如何使用它。

loadPlugin, executePluginCommand

您可以使用加载插件命令来加载一个PyBullet插件：

| | | | |
|------|-----------------|-----|--------------------------|
| 必须的 | pluginPath | 细绳 | 路径，在磁盘上找到插件的位置 |
| 必须的 | 后修复 | 细绳 | 附加到每个API中的插件的后缀名称 |
| 可选择的 | physicsClientId | int | 如果您已连接到多个服务器，则您可以选择一个服务器 |

加载插件将返回一个插件UniqueId整数。如果这个插件id是负的，那么这个插件就不是载重的一旦加载了一个插件，您就可以使用它向该插件发送命令

executePluginCommand:

| | | | |
|------|-----------------|-------|--------------------------|
| 必须的 | pluginUniqueId | int | 插件的唯一id，由加载插件返回 |
| 可选择的 | 文本参数 | 细绳 | 可选的文本参数，将由插件进行解释 |
| 可选择的 | intArgs | int列表 | 可选的整数列表，可由插件进行解释 |
| 可选择的 | floatArgs | 浮子列表 | 可选的浮点数列表，可由插件进行解释 |
| 可选择的 | physicsClientId | int | 如果您已连接到多个服务器，则您可以选择一个服务器 |

unloadPlugin

你可以使用这个插件id来卸载一个插件。

该插件API与PyBullet共享相同的底层C API，并且与PyBullet具有相同的特性。

你可以浏览这个插件[实现的小球，以了解什么是可能的。](#)

构建和安装PyBullet

有几种不同的方法可以在窗口、Mac OSX和Linux上安装小子弹。我们使用Python 2.7和Python 3.5.2，但预计大多数都是Python 2.x和Python3.x版本应该可以工作。

最容易让小子弹工作是使用pip或pythonsetut.py:

使用Python pip

确保安装了Python和pip，然后运行:

pip安装弹丸

您可能需要使用`sudo pip`安装项目项目或`pip`安装项目项目用户。

注意，如果您使用`pip`安装PyBullet，那么安装C++子弹物理SDK仍然有益的：它包括数据文件、物理服务器和对PyBullet有用的工具。

您还可以运行“python设置”。“py构建”和“python设置”。[在子弹物理SDK的根目录中安装](http://bulletphysics.com/bullet3) ‘（从<http://github.com/bulletphysics/bullet3>获得SDK）

参见<https://pypi.org/project/pybullet>

或者，您可以使用预制作（Windowe）或`cmake`从源代码安装PyBullet

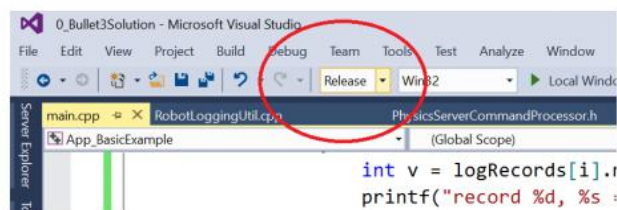
确保在`c:\python-3.5.2`中安装了某些Python版本（或其他版本的文件夹名称）

首先从github中获取源代码，使用

[git克隆https://github.com/bulletphysics/bullet3](https://github.com/bulletphysics/bullet3)

点击`build_visual_studio_vr_pybullet_double`上。蝙蝠，并打开`0_Bullet3Solution.sln`项目在Visual Studio中，转换项目，如果需要。

切换到释放模式，并编译“小副项目符号”项目。



然后有一些选项可以在Python解释器中导入小符号：

- 1) 重命名`pybullet_vs2010.dll`到`pybullet.pyd`，并启动`Python.exe`解释器使用作为当前的工作目录。可选择调试：重命名`metel/bin/pybullet_vs2010_debug.dll` `pybullet_d.pyd`并启动`python_d.exe`
- 2) 重命名项目符号`/bin/pybullet_vs2010.dll`到`pybuner`。使用命令提示符：设置蟒蛇路径`c:\开发子弹3\bin`（替换为实际文件夹）或使用窗口GUI创建这个蟒蛇路径环境变量
- 3) 创建一个管理员提示符(cmd)。并创建如下所示的符号链接

```
cd c:\python-3.5.2\dlls
```

```
mklink pybullet.pyd c:\develop\bullet3\bin\pybullet_vs2010.dll
```

然后运行python。exe和导入小子弹应该工作。

在Linux和Mac OSX上使用cmake

请注意，建议的方法是使用sudo pip安装小项目符号（或pip3）。使用cmake或premake或其他构建系统只适用于那些知道自己在做什么的开发人员，而且通常不受支持。

首先从github中获取源代码，使用

git克隆<https://github.com/bulletphysics/bullet3>

1) 下载并安装cmake

2) 在Bullet:

build_cmake_pybullet_double的根目

录中运行shell脚本。sh

3) 确保Python找到我们的小子弹。所以模块:

导出通通= /your_path_to_bullet/build_cmake/examples/pybullet

就是这样。通过运行python解释器来测试小项目，并输入“导入小项目项目”，以查看模块是否加载。如果是这样，您可以在子弹/示例/子弹中使用子弹脚本。

可能的Mac OSX问题

- 如果您在Mac OSX上导入小符号有任何问题，请确保运行正确的Python解释器，匹配-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，例如在使用自制程序时。[看到这个 评论一个例子。](#)
- 尝试使用CFLAGS= '-stdlib=libc++' pip安装项目符号，请看这个问题。

可能的Linux问题

- 请确保已安装了OpenGL
- 当使用蟒蛇作为Python发行版时，请安装libgcc，以便找到“glibcxx”（见<http://askubuntu.com/questions/575505/glibcxx-3-4-20-not-found-how-to-fix-this-error>）
- 当使用蟒蛇作为python发行版时，cmake可能无法找到蟒蛇库。您可以通过进入../build_cmake/CMakeCache来手动添加它们。txt文件和更改以下行：
“PYTHON_LIBRARY: FILEPATH=/usr/lib/python2.7/config-x86_64-linux-gnu/libpython2.7. 如此

GPU或虚拟机缺少OpenGL 3

- 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，这会导致工件（灰色屏幕）甚至崩溃。您可以使用——OpenGL 2标志返回到OpenGL 2。这并没有完全支持，但它提供了一些查看场景的方式。：

○ pybullet.连接（弹丸。GUI，选项=“-opengl2”）

- 或者，您可以在远程计算机上使用UDP或TCP桥运行物理服务器，并通过UDP隧道从本地笔记本电脑连接到远程服务器。

（任务：详细描述步骤）

支持、提示、引文

问：我们应该去哪里寻求支持和报告问题？

答案：[有一个讨论论坛在http://pybullet.org/Bullet](http://pybullet.org/Bullet)和一个[问题跟踪器在https://github.com/bulletphysics/bullet3](https://github.com/bulletphysics/bullet3)

问：我们如何在我们的学术论文中添加一个引用到小子弹？

答案：`@MISC{coumans2020,
 作者=, 欧文·库曼斯和白云飞},
 标题={小子弹, 一个用于游戏、机器人和机器学习物理模拟的Python模块},
 howpublished = {\url{http://pybullet.org}},
 =年{2016-2020}
}`

问：小子弹可以在谷歌Colab中使用吗？

答案：是的，我们提供了预编译的手动linux轮，可以用于Colab。GPY渲染也可以使用EGL进行工作。这里是Colab的一个例子。[子弹2到底怎么了。x和Bullet 3的OpenCL实现吗？](#)

问：PyBullet正在包装[子弹 capi](#)。我们将把Bullet3OpenCLGPUAPI（和未来的Bullet 4。在这个C-API的背后。所以，如果你使用PyBullet或C-API，你是可以证明未来的。不要和弹头2混淆。xC++API。

问：我是否应该使用扭矩/力控制或速度/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/扭矩控制能够可靠地工作，还需要付出更多的努力。

问：物体的速度似乎比预期的要小。小子弹是否应用了一些默认的阻尼？而且，速度也不超过100个单位。

答：是的，小子弹应用了一些角度和线性阻尼来增加稳定性。你

可以使用“变化动态”命令修改/禁用这个阻尼，使用线性阻尼=0和角度阻尼=0作为参数。

最大线/角速度固定在100单位以保持稳定。

问：如何只关闭机器人的某些部分（比如手臂）关闭重力？

答案：

目前这还没有暴露，所以你需要对所有对象旋转重力加速度，并对需要它的对象手动应用重力。或者，你也可以主动计算重力补偿力，就像在一个真实的机器人上发生的情况一样。由于子弹有一个完整的约束系统，计算这些反重力将是微不足道的：你可以运行第二个模拟（PyBullet让你连接到多个物理服务器）和定位机器人在重力下，设置联合位置控制以保持需要的位置，并收集这些“反重力”力。然后将其应用于主仿真中。

问题：如何放大/缩小物体？

答：您可以使用全局比例因子值作为可选参数来加载URDF和

loadSDF。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，如URDF和SDF。目前，您还不能重新缩放对象。

问：我如何在我的模型中获得纹理？

答：您可以使用波前文件。obj文件格式。这将支持材料文件（.mtl）。

在子弹/数据文件夹中有各种使用纹理的示例。您可以使用“更改纹理”API来更改现有纹理对象的纹理。

问：哪些纹理文件格式对PyBullet有效？

答：子弹使用stb_image加载纹理文件，它加载PNG、JPG、TGA、GIF等。

[看到stb_image.h](#)的细节。

问：如何提高碰撞检测的性能和稳定性？

答：有很多可以进行优化的方法，例如：

形状类型

1) 选择一个或多个原始碰撞形状类型，如框、球体、胶囊、圆柱体来近似一个对象，而不是使用凸或凹三角形网格。

2) 如果您真的需要使用三角形网格，请使用层次近似凸分解（v-HACD）创建一个凸分解。

这个[test_hacd](#)实用程序将OBJ文件中的凸三角形网格转换为具有多个凸包对象的新的OBJ文件。请参阅例如[Bullet/data/teddy_vhacd.urdf](#)指向子弹头
/data/teddy2_VHACD_CHs.obj或[duck_vhacd.urdf](#)指向duck_vhacd.obj。

3) 减少三角形网格中的顶点数量。例如，Blender 3D有一个很好的网格抽取修改器，它可以交互式地让您看到网格简化的结果。

- .014) 使用小的正值的滚动摩擦（例如0）和旋转摩擦的圆形物体，如球体和胶囊和机器人抓手使用<滚动摩擦>和<旋转摩擦>节点在<链接><接触>节点。请参见，例如
Bullet/data/sphere2.urdf
- 5) 使用URDF<链接><接触>xml节点内的<刚度值==1000/>对车轮使用少量的符合性。例如
Bullet/data/husky/husky.urdf车辆。
- 6) 使用的双精度构建子弹，这对接触稳定性和碰撞精度都很好。选择一些好的约束求解器设置和时间步长。
- 7) 从图形中解耦物理模拟。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。

问：摩擦处理的选择是什么？

答：默认情况下，子弹和小球子弹对库仑摩擦模型使用精确的隐式锥摩擦。此外，您可以通过在<链接><联系人>节点中添加一个<滚动摩擦>和<旋转摩擦>节点来启用滚动和旋转摩擦，请参阅子弹/数据/球体2。例如urdf。可以启用锥近似，而不是锥摩擦。

问：子弹内部是什么样的恒定或阈值，这使得高速不可能实现？答：默认情况下，子弹依赖于离散碰撞检测和穿透恢复。纯粹依赖于离散的碰撞检测意味着一个物体在一个时间步长内的移动速度不应该超过它自己的半径。PyBullet使用1./240作为默认的时间步长。时间步长大于1./60可能会导致各种原因的不稳定（深度渗透、数值积分器）。子弹是一个连续碰撞检测的选项，以捕捉在一个时间步内移动速度超过其自己的半径的物体的碰撞。不幸的是，这种连续的碰撞检测可能会引入它自身的问题（性能和非物理响应，缺乏恢复），因此在默认情况下不启用这个实验特性。[请查看](#)
[“experimentalCcdSphereRadius.”](#)例如，如何启用它。

问：有些api没有被文档记录下来。通常这意味着(1)我们还没有更新快速入门指南，或者(2)该功能太实验性，无法记录。如果您真的想知道一个特定的未文档记录的API，您可以在跟踪器中提交一个问题。