Digital-Twin

Table of Contents

1 概述

本文的目标在于梳理并定义前端业务接口规范，用于避免后端开发者在不完全了解前端业务的情况下开发出不兼容的接口。

1.1 安装

建立源代码目录：`mkdir -p ./GraspSim && cd GraspSim`

下载源代码：`git clone http://120.24.55.96:3000/MixedAI/GraspSim`

安装依赖：`pip install -r requirements.txt`

1.2 快速开始

获取一堆工件的深度图：`python3 test.py`

获取一堆工件的姿态数据：`python3 test0.py`

混合拆垛演示：`python3 test1.py`

无序抓取演示：`python3 test2.py`

2 工作计划

目标	任务	问题	备注
接口定义	场景查看	载入/重置	
		获取物体标识	
		获取场景画面	启动工作流后卡顿
		暂停/继续仿真	
		平移/旋转/缩放/焦点	
		视图切换-前后左右	
		绘制原点/碰撞点	实现较为复杂
		关闭场景	前端窗口无法正确析构
	场景编辑	坐标检测	
		选择/添加/删除/保存	
		透明化	
		移动/旋转/缩放	
	物体	位置/姿态	
		标识	
		模型更换	
		纹理更换	
	物体.机械臂	获取/设置末端执行器	
		获取/设置末端执行器姿态	
		获取/设置末端执行器位置	
		设置末端执行器数字输出	
		获取关节数	
		获取/设置关节位置	
		获取/设置速度	

目标	任务	问题	备注
		设置当前位置为Home点	
		前往Home点	
	物体.相机	获取相机画面	
		获取/设置标定参数	
		设置/清除点云	
	物体.码垛器	区域参数	
		工件更换	
	物体.放置器	区域参数	
		获取/设置工件	
		获取/设置工件纹理	
		获取/设置放置点	
		获取/设置总计工件数	
		获取/设置缩放因子	
		获取/设置放置模式	
	工作流	获取可用节点	
		设置/获取	
		启动/停止	
	....	....	
SimEngine 陈君辉	验收要求	载入物体的示例	可以调节位置和姿态，并且与地面发生物理交互，如碰撞
		获取RGBD图的示例	显示RGB图，深度图展现为灰度图的形式
		获取物体姿态的示例	通过射线检测物体的存在，然后把该物体的姿态轴画出来即可
		吸盘吸附物体的示例	对力的控制功能

目标	任务	问题	备注
		夹爪拾取物体的示例	对摩擦的控制功能
		控制一个关节的运动，使另一个关节进行同步运动	对关节的同步控制功能
	....		
DexSim 李瑶	验收要求	一堆工件生成的示例	
		一堆工件的深度图示例	
		一堆工件的姿态示例	
		控制机械臂+吸盘去拾取工件	
		控制机械臂+夹爪去拾取工件	IK驱动机械臂

3 业务流程

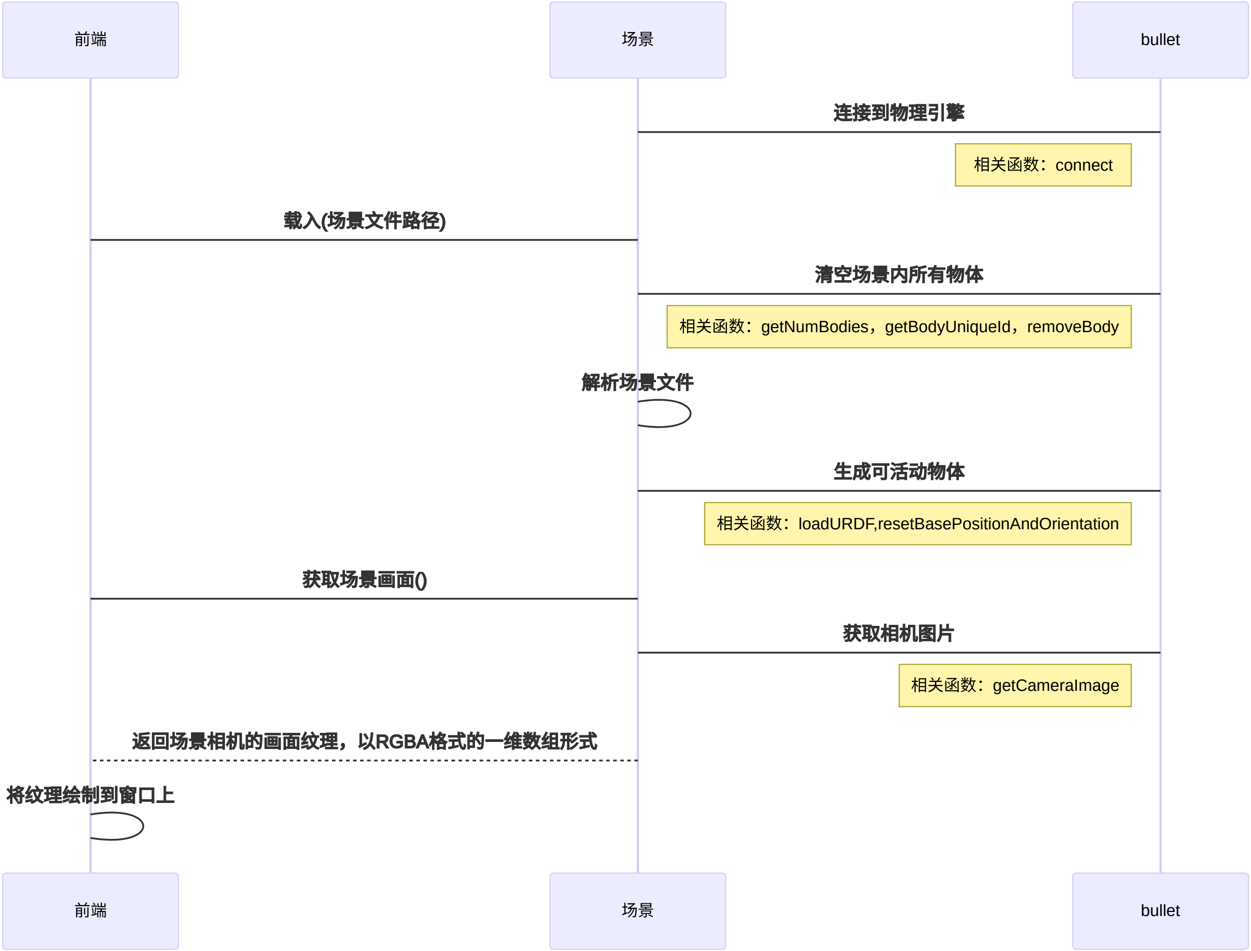
3.1 场景-载入场景并获取画面

通过解析一个场景文件并加载其定义的物体到分拣场景中，通过渲染到纹理技术获取场景画面到前端显示。

文件定义如下：

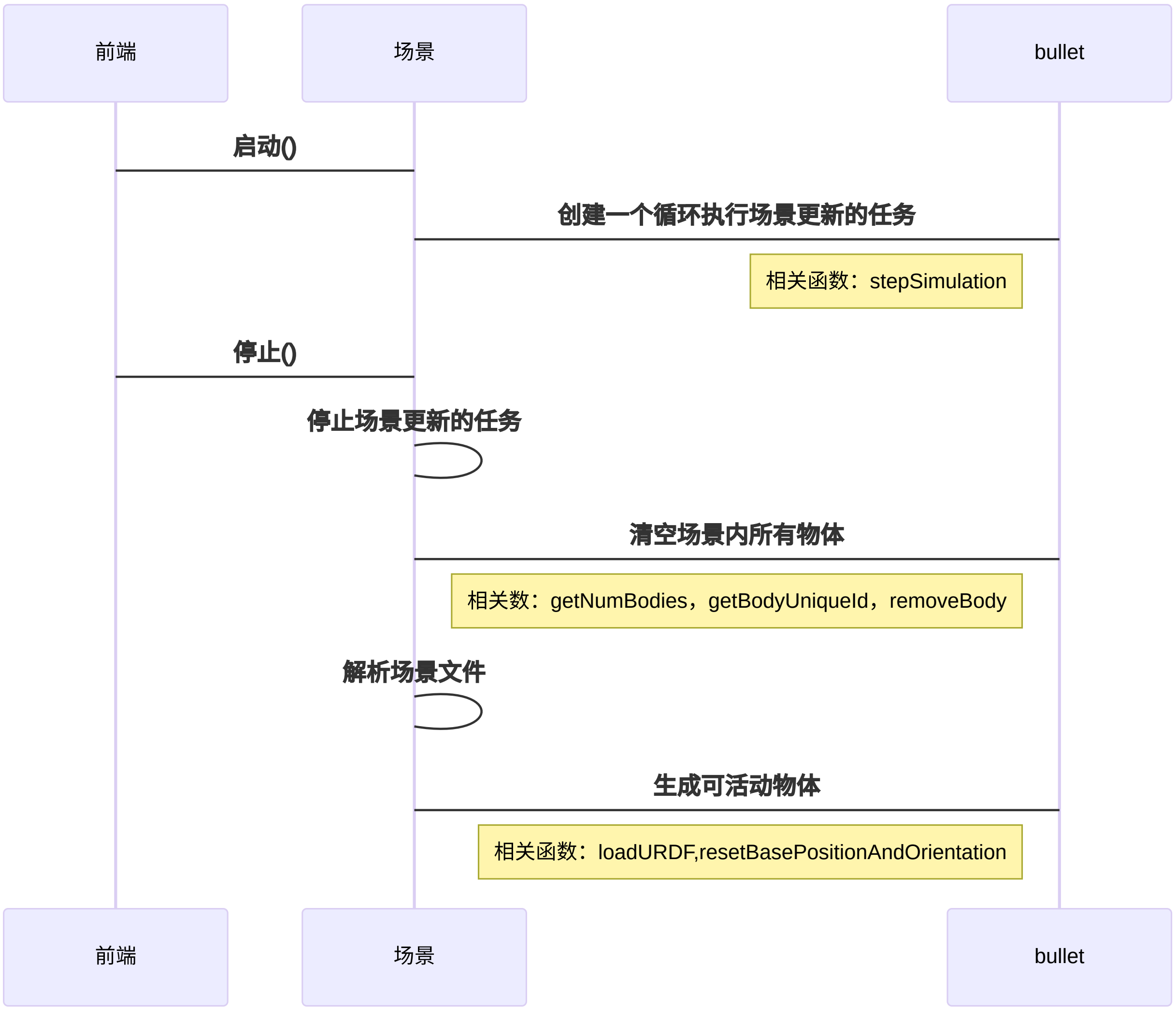
```
scene_profile = {
    "active_objects": [
        {
            "kind":"Robot",
            "name":"robot",
            "base":"./data/robots/ur5.urdf",
            "pos":[0,0,0],
            "rot":[0,0,1.57],
            "end_effector":"./data/end_effectors/magnet.urdf"
        },
        {
            "kind":"Placer",
            "name":"placer",
            "base":"./data/objects/tray/traybox.urdf",
            "pos":[0,-0.5,0.001],
            "rot":[0,0,0],
            "center":[0.0,-0.5,0.25],
            "interval":0.1,
            "amount":30,
            "workpiece":"./data/workpieces/lego/lego.urdf"
        },
        {
            "kind":"Camera3D",
            "name":"camera",
            "base":"./data/cameras/camera3d.urdf",
            "pos":[-0.5,-0.5,0.0],
            "rot":[0.0,0.0,-1.57],
            "fov": 45,
            "forcal": 0.01,
            "image_size": [300,300],
            "image_path":"./guagua.png"
        }
    ],

    "workflow": {...} # 工作流
}
```



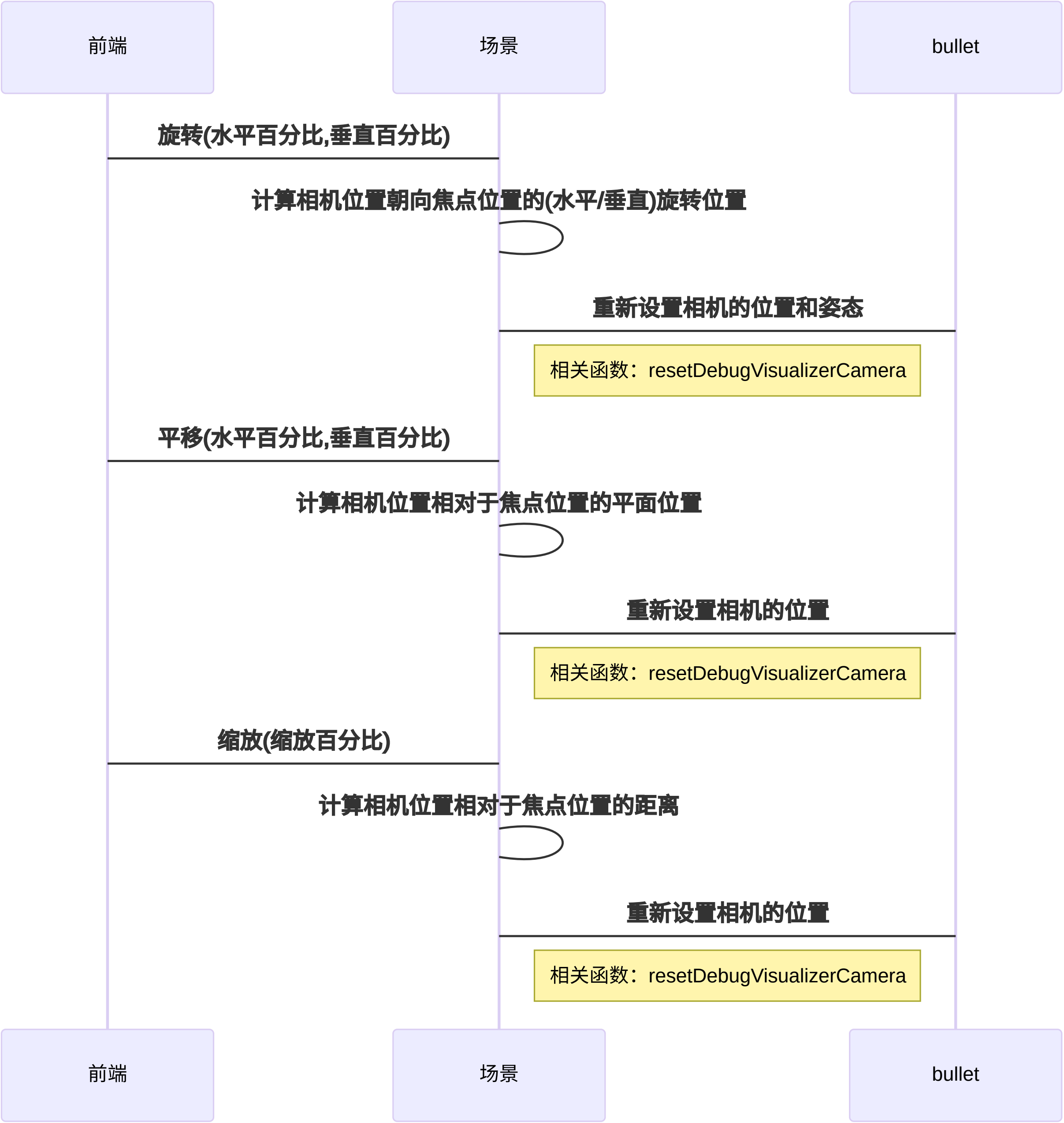
3.2 场景-启动/停止

当场景布置完毕后，执行启动即可持续进行物理运算。停止则暂停物理运算。


前端
场景
bullet
启动()
创建一个循环执行场景更新的任务
相关函数：stepSimulation
停止()
停止场景更新的任务
清空场景内所有物体
相关数：getNumBodies，getBodyUniqueId，removeBody
解析场景文件
生成可活动物体
相关函数：loadURDF,resetBasePositionAndOrientation
前端
场景
bullet

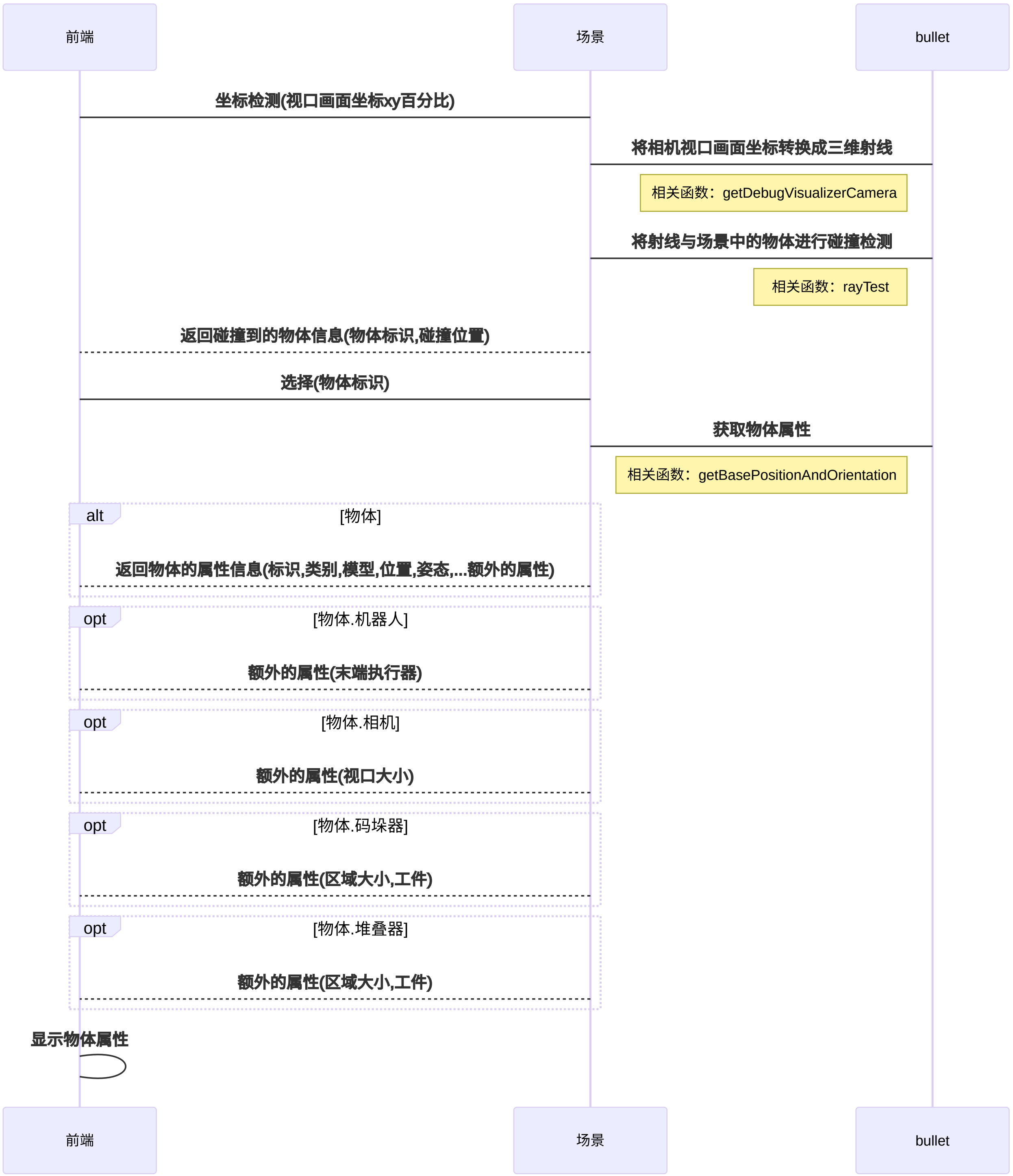
3.3 场景-视口控制

用户会经常使用鼠标或者手势进行镜头控制。


前端
场景
bullet
旋转(水平百分比,垂直百分比)
计算相机位置朝向焦点位置的(水平/垂直)旋转位置
重新设置相机的位置和姿态
相关函数：resetDebugVisualizerCamera
平移(水平百分比,垂直百分比)
计算相机位置相对于焦点位置的平面位置
重新设置相机的位置
相关函数：resetDebugVisualizerCamera
缩放(缩放百分比)
计算相机位置相对于焦点位置的距离
重新设置相机的位置
相关函数：resetDebugVisualizerCamera
前端
场景
bullet

3.4 场景-获取选中物体的信息

用户在布置场景的时候会通过鼠标获取物体的各种信息。

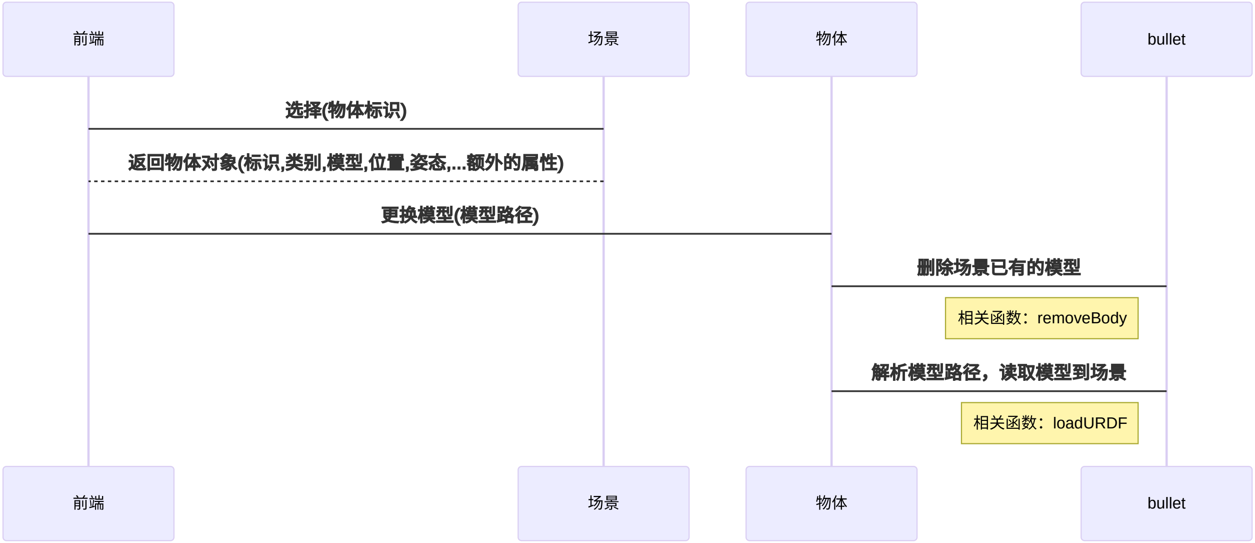

前端
场景
bullet
坐标检测(视口画面坐标xy百分比)
将相机视口画面坐标转换成三维射线
相关函数：getDebugVisualizerCamera
将射线与场景中的物体进行碰撞检测
相关函数：rayTest
返回碰撞到的物体信息(物体标识,碰撞位置)
选择(物体标识)
获取物体属性
相关函数：getBasePositionAndOrientation
alt
[物体]
返回物体的属性信息(标识,类别,模型,位置,姿态,...额外的属性)
opt
[物体.机器人]
额外的属性(末端执行器)
opt
[物体.相机]
额外的属性(视口大小)
opt
[物体.码垛器]
额外的属性(区域大小,工件)
opt
[物体.堆叠器]
额外的属性(区域大小,工件)
显示物体属性
前端
场景
bullet

3.5 场景-保存

当对场景进行了编辑之后，调用此函数来进行保存。

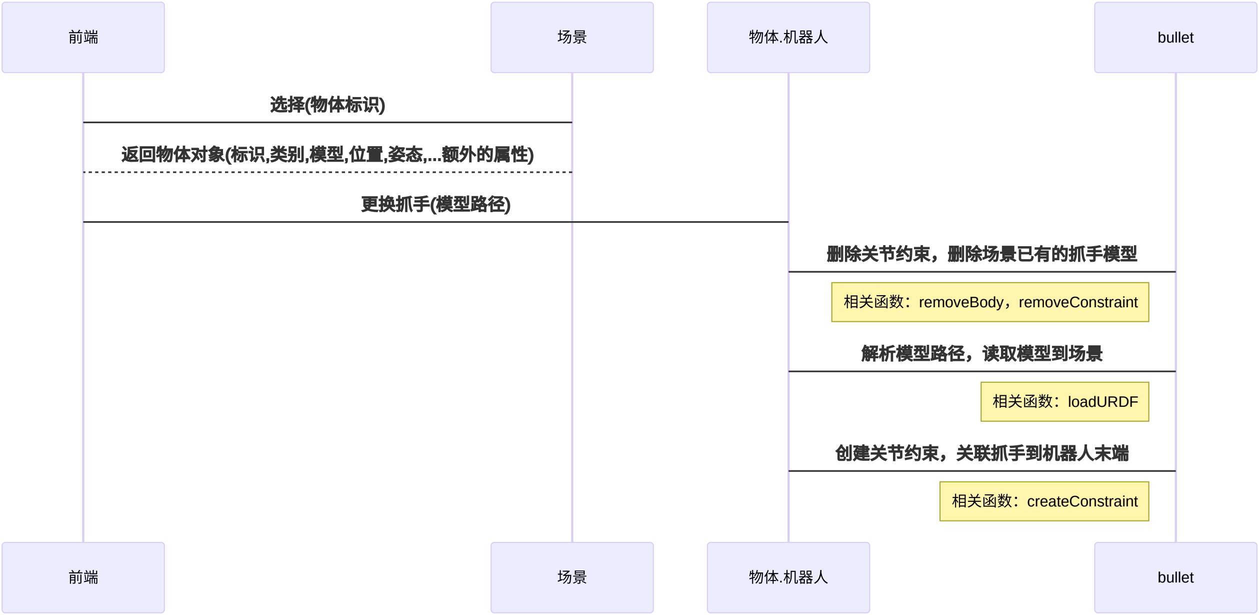
3.5 物体-更换模型

用户布置场景的时候会对模型进行替换。



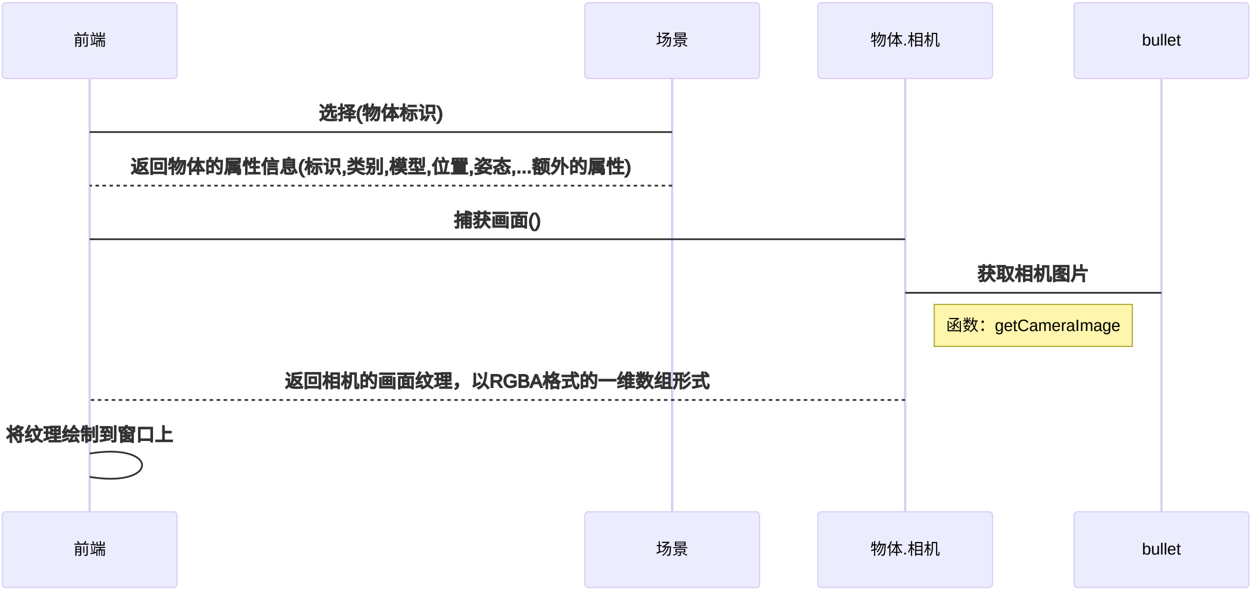
3.6 物体.机器人-更换抓手

用户在为不同工件采用不同的机器人抓手。


前端
场景
物体.机器人
bullet
选择(物体标识)
返回物体对象(标识,类别,模型,位置,姿态,...额外的属性)
更换抓手(模型路径)
删除关节约束，删除场景已有的抓手模型
相关函数：removeBody，removeConstraint
解析模型路径，读取模型到场景
相关函数：loadURDF
创建关节约束，关联抓手到机器人末端
相关函数：createConstraint
前端
场景
物体.机器人
bullet

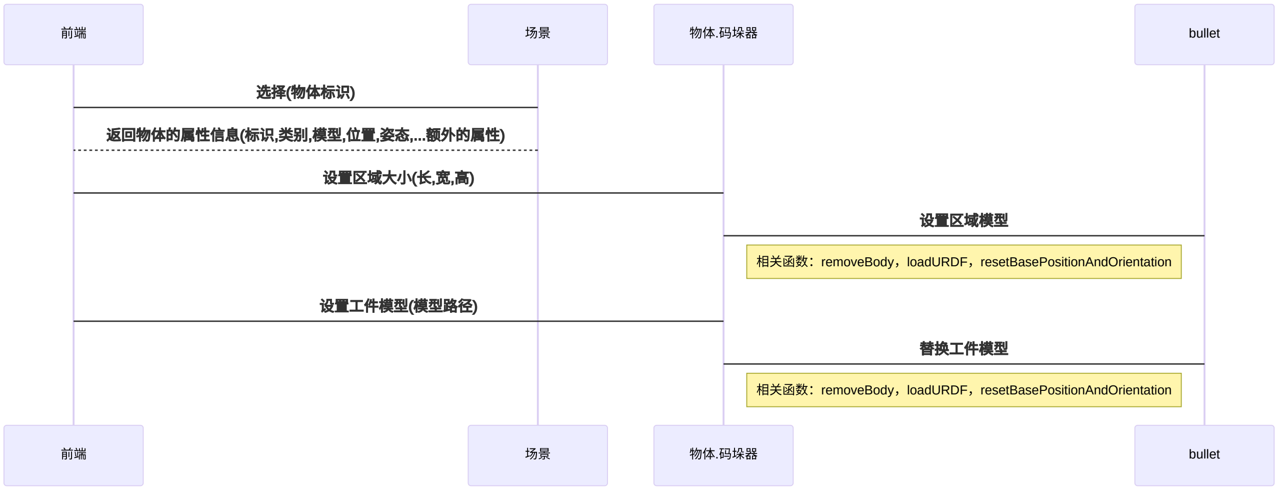
3.7 物体.相机-捕获画面

虚拟相机的拍照功能。



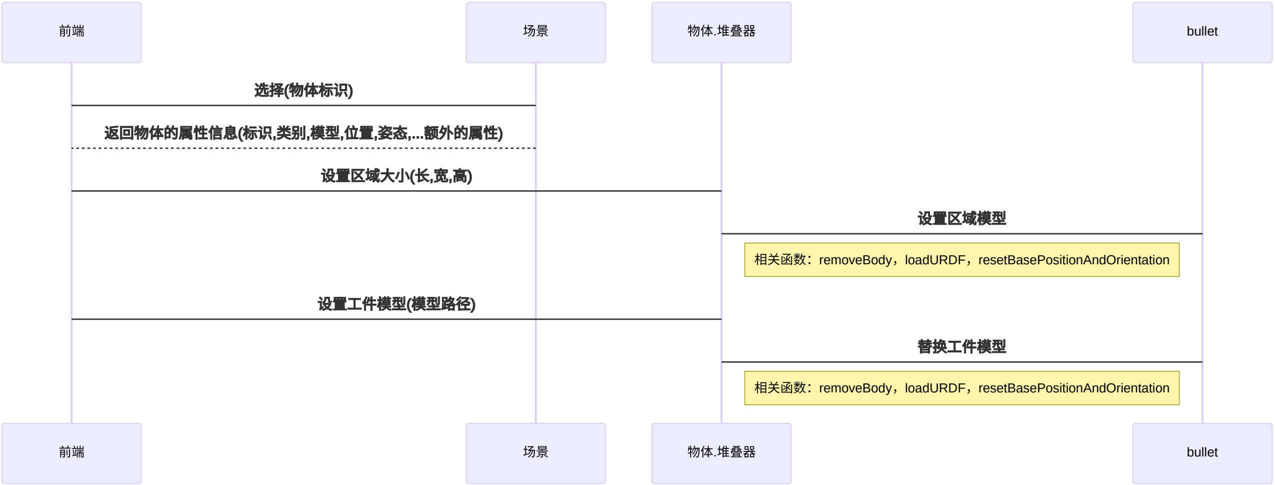
3.8 物体.码垛器

用于生成码放有序的工件。


前端
场景
物体.码垛器
bullet
选择(物体标识)
返回物体的属性信息(标识,类别,模型,位置,姿态,...额外的属性)
设置区域大小(长,宽,高)
设置区域模型
相关函数：removeBody，loadURDF，resetBasePositionAndOrientation
设置工件模型(模型路径)
替换工件模型
相关函数：removeBody，loadURDF，resetBasePositionAndOrientation
前端
场景
物体.码垛器
bullet

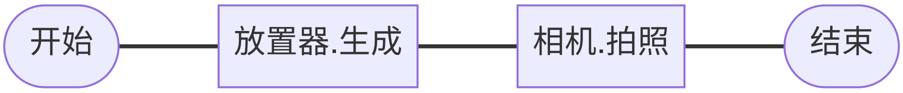
3.9 物体.放置器

用于生成码放无序的工件。


前端
场景
物体.堆叠器
bullet
选择(物体标识)
返回物体的属性信息(标识,类别,模型,位置,姿态,...额外的属性)
设置区域大小(长,宽,高)
设置区域模型
相关函数：removeBody，loadURDF，resetBasePositionAndOrientation
设置工件模型(模型路径)
替换工件模型
相关函数：removeBody，loadURDF，resetBasePositionAndOrientation
前端
场景
物体.堆叠器
bullet

3.10 工作流-示例-深度图/姿态估计/混合拆垛/无序抓取

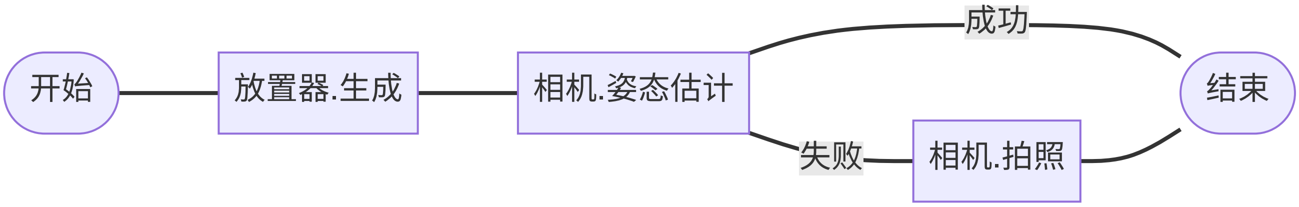
通过简单的两个节点来实现获取深度图的工作流，如下：



```
scene_profile = {
  "active_objects": [{
      "kind":"Placer",
      "name":"placer",
      "base":"./data/objects/tray/traybox.urdf",
      "pos":[0,-0.5,0.001],
      "rot":[0,0,0],
      "center":[0.0,-0.5,0.25],
      "interval":0.1,
      "amount":30,
      "workpiece":"./data/workpieces/lego/lego.urdf"
    },{
      "kind":"Camera3D",
      "name":"camera",
      "base":"./data/cameras/camera3d.urdf",
      "pos":[-0.5,-0.5,0.0],
      "rot":[0.0,0.0,-1.57],
      "fov": 45,
      "forcal": 0.01,
      "image_size": [300,300],
      "image_path":"./guagua.png"
    }
  ],

  "workflow":{
    "run":"1",
    "declare":{
      "1":{"kind":"Packer","fun":"generate","name":"placer","next":"2"},
      "2":{"kind":"Camera","fun":"capture","name":"camera"}
    }
  }
}
```

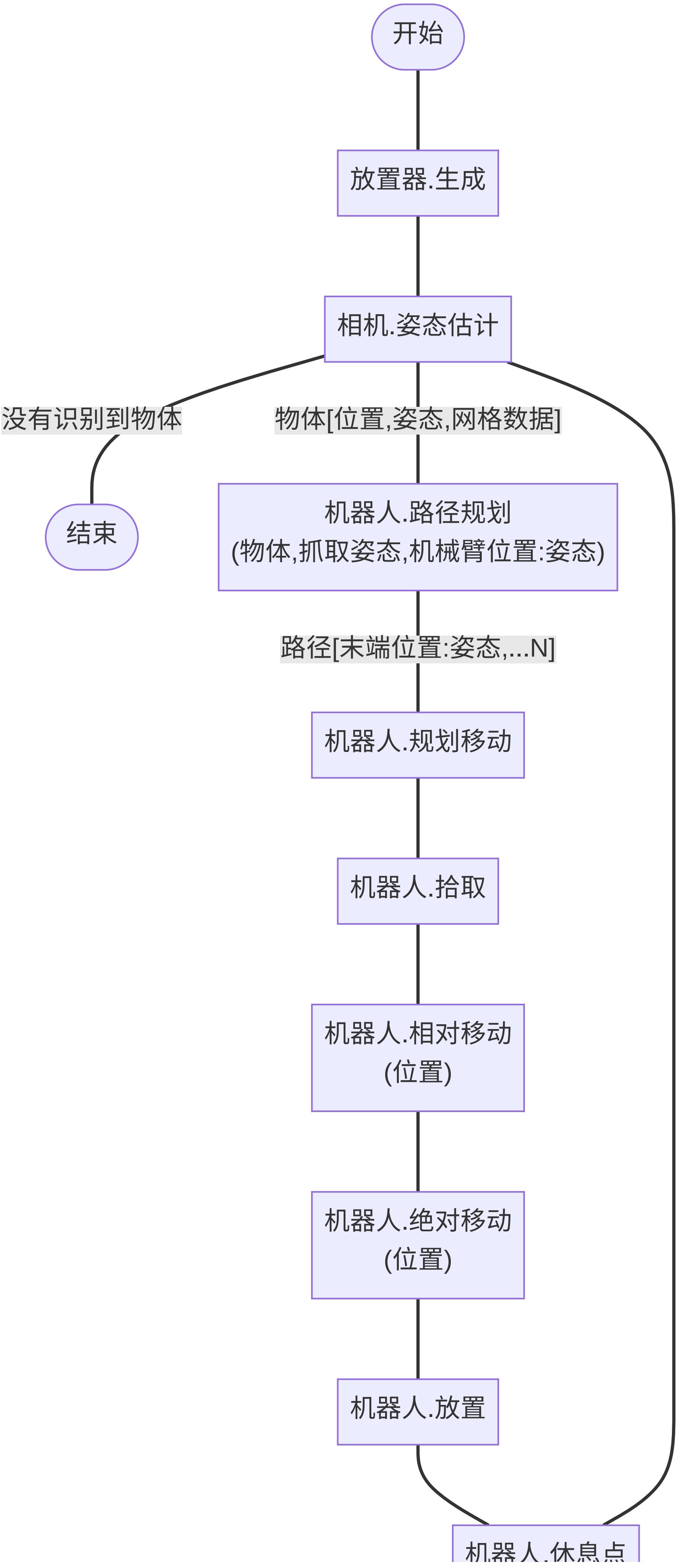
节点有时因为一些限制参数会返回失败，为应对于不同情况，可以采用分支来进行控制。假设工件生成成功，就进行姿态估计，否则就拍照，如下：



```
scene_profile = {
  "active_objects": [
    {
      "kind":"Placer",
      "name":"placer",
      "base":"./data/objects/tray/traybox.urdf",
      "pos":[0,-0.5,0.001],
      "rot":[0,0,0],
      "center":[0.0,-0.5,0.25],
      "interval":0.1,
      "amount":10,
      "workpiece":"./data/workpieces/lego/lego.urdf"
    },
    {
      "kind":"Camera3D",
      "name":"camera",
      "base":"./data/cameras/camera3d.urdf",
      "pos":[-0.5,-0.5,0.0],
      "rot":[0.0,0.0,-1.57],
      "fov": 20,
      "forcal": 0.01,
      "sample_rate": 20,
      "image_size": [300,300],
      "image_path":"./guagua.png"
    }
  ],

  "workflow":{
    "run":"1",
    "declare":{
      "1":{"kind":"Placer","fun":"generate","name":"placer","next":"2"},
      "2":{"kind":"Camera","fun":"pose_recognize","name":"camera",
        "alt":[
          {"next":"3","err":"failed"}
        ]
      },
      "3":{"kind":"Camera","fun":"capture","name":"camera"}
    }
  }
}
```

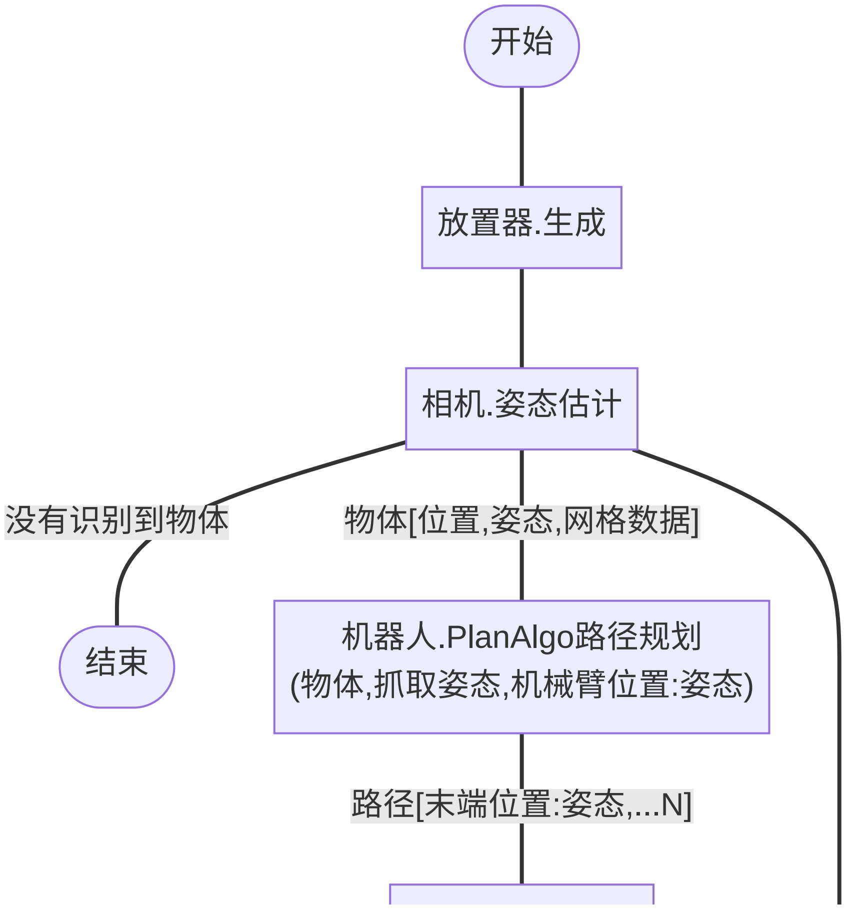
接着，设计一个混合拆垛工作流，加入机器人进行分拣工作，如下：

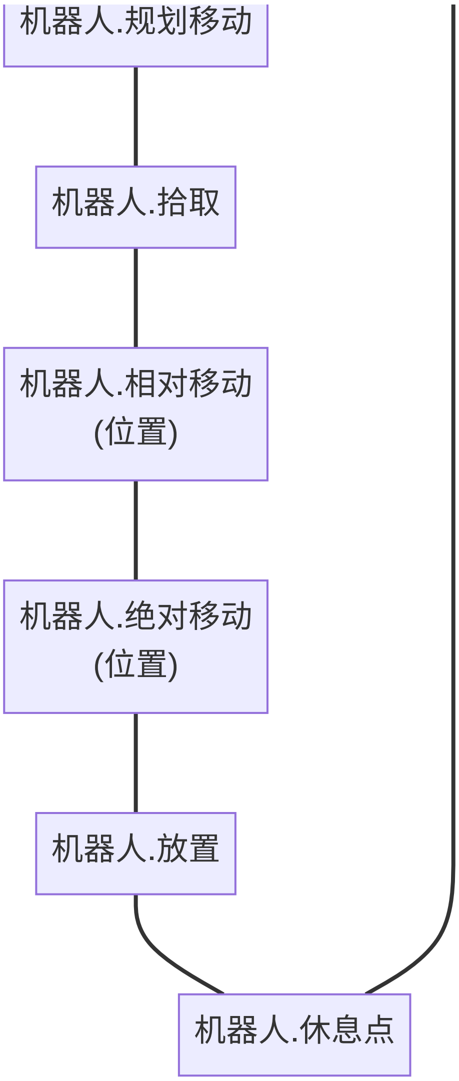

开始
放置器.生成
相机.姿态估计
没有识别到物体
物体[位置,姿态,网格数据]
结束
机器人.路径规划
(物体,抓取姿态,机械臂位置:姿态)
路径[末端位置:姿态,...N]
机器人.规划移动
机器人.拾取
机器人.相对移动
(位置)
机器人.绝对移动
(位置)
机器人.放置
机器人.休息点

```
# 格式：json
scene_profile={
  "active_objects": [
    {
      "kind":"Robot",
      "name":"robot",
      "base":"./data/robots/ur5/ur5.urdf",
      "pos":[0,0,0],
      "rot":[0,0,3.14],
      "reset_joint_poses":[-1.57,0.0,-0.3925,-0.785,1.57,0],
      "joint_damping":[ 0, 1, 0.9, 0.8, 0.7, 0.0],
      "end_effector":"./data/end_effectors/suction/suction.urdf"
    },
    {
      "kind":"Stacker",
      "name":"stacker",
      "base":"./data/objects/tray/traybox.urdf",
      "pos":[0,-0.52,0.01],
      "rot":[0,0,0],
      "area":[0.3,0.3,0.3],
      "box_size":[0.1,0.1,0.05],
      "random_factor":[0.2,0.2,0.0]
    },
    {
      "kind":"Camera3D",
      "name":"camera",
      "base":"./data/cameras/camera3d.urdf",
      "pos":[-0.5,-0.52,0.0],
      "rot":[0.0,0.0,-1.57],
      "fov": 20,
      "forcal": 0.01,
      "sample_rate": 20,
      "image_size": [300,300],
      "image_path":"./guagua.png"
    }
  ],

  "workflow": {
    "run":"1",
    "declare":{
      "1":{"kind":"Stacker","fun":"generate","name":"stacker", "next":"2"},
      "2":{
        "kind":"Camera3D","fun":"pose_recognize","name":"camera", "next":"3",
        "alt":[
          {"next":"7","err": "failed"}
        ]
      },
      "3":{"kind":"Robot","fun":"pick_plan","name":"robot","next":"4","args":{
        "pick_poses":[
          {"pos":[0.0,0.0,0.024],"rot":[0,0,0]}
        ]
      }},
      "4":{"kind":"Robot","fun":"plan_move","name":"robot","next":"5"},
      "5":{"kind":"Robot","fun":"do","name":"robot","args":{"pickup": true},"next":"6"},
      "6":{"kind":"Robot","fun":"move_relatively","name":"robot","args":{"x":0.0,"y":0.0,"z":0.1},"next":"7"},
      "7":{"kind":"Robot","fun":"move","name":"robot","args":{"x":0.5,"y":0.0,"z":0.5},"next":"8"},
      "8":{"kind":"Robot","fun":"do","name":"robot","args":{"pickup": false},"next":"9"},
      "9":{"kind":"Robot","fun":"home","name":"robot","next":"2"}
    }
  }
}
```

最后，设计一个无序抓取的工作流，因为内置算法太垃圾，在此加入第三方路径规划算法，如下：




机器人.规划移动
机器人.拾取
机器人.相对移动(位置)
机器人.绝对移动(位置)
机器人.放置
机器人.休息点

```
scene_profile={
  "active_objects": [
    {
      "kind":"Robot",
      "name":"robot",
      "base":"./data/robots/ur5/ur5.urdf",
      "pos":[0,0.1,0],
      "rot":[0,0,3.14],
      "reset_joint_poses":[-1.57,0.0,-0.3925,-0.785,1.57,0],
      "joint_damping":[ 0, 1, 0.9, 0.8, 0.7, 0.0],
      "end_effector":"./data/end_effectors/gripper/gripper.urdf"
    },
    {
      "kind":"Placer",
      "name":"placer",
      "base":"./data/objects/tray/traybox.urdf",
      "pos":[0,-0.5,0.001],
      "rot":[0,0,0],
      "center":[0.0,-0.5,0.25],
      "interval":0.1,
      "amount":10,
      "workpiece":"./data/workpieces/suction/suction.urdf",
      "workpiece_texture":""
    },
    {
      "kind":"Camera3D",
      "name":"camera",
      "base":"./data/cameras/camera3d.urdf",
      "pos":[-0.5,-0.5,0.0],
      "rot":[0.0,0.0,-1.57],
      "fov": 20,
      "forcal": 0.01,
      "sample_rate": 20,
      "image_size": [300,300],
      "image_path":"./guagua.png"
    }
  ],

  "workflow":{
    "run":"1",
    "declare":{
      "1":{"kind":"Placer","fun":"generate","name":"placer", "next":"2"},
      "2":{
        "kind":"Camera3D","fun":"pose_recognize","name":"camera", "next":"3",
        "alt":[{"next":"7","err": "failed"}]
      },
      "3":{"kind":"Robot","fun":"pick_plan","name":"robot","args":{
        "pick_poses":[
          {"pos":[0.0,0.0,0.01],"rot":[0,1.57,0]},
          {"pos":[0.0,0.0,0.01],"rot":[0,1.57,-0.785]},
          {"pos":[0.0,0.0,0.01],"rot":[0,1.57,-1.57]},
          {"pos":[0.0,0.0,0.01],"rot":[0,1.57,-2.355]},
          {"pos":[0.0,0.0,0.01],"rot":[0,1.57,-3.14]},
          {"pos":[0.0,0.0,0.01],"rot":[0,1.57,2.355]},
          {"pos":[0.0,0.0,0.01],"rot":[0,1.57,1.57]},
          {"pos":[0.0,0.0,0.01],"rot":[0,1.57,0.785]}
        ]
      },"next":"4"},
      "4":{"kind":"Robot","fun":"plan_move","name":"robot","next":"5"},
      "5":{"kind":"Robot","fun":"do","name":"robot","args":{"pickup":true},"next":"6"},
      "6":{"kind":"Robot","fun":"move_relatively","name":"robot","args":
{"x":0.0,"y":0.0,"z":0.35},"next":"7"},
      "7":{"kind":"Robot","fun":"move","name":"robot","args":{"x":0.3,"y":0.1,"z":0.3},"next":"8"},
      "8":{"kind":"Robot","fun":"do","name":"robot","args":{"pickup": false},"next":"9"},
      "9":{"kind":"Robot","fun":"home","name":"robot","next":"2"}
    }
  }
}
```

3.11 工作流-获取活动节点

绘制流程图前，前端需要知道流程图由多少种节点构成，通过这个函数可以得到场景中所有可用的节点及功能，并用名字区分实体。

```
#输入
workflow.get_active_obj_nodes(
  #无参
)\n

#输出
[
  {
    "kind": "Robot", #机器人节点
    "funs": [  #可用功能
      {"f":"move","errs":[]},
      {"f":"pick","errs":[]},
      {"f":"place","errs":[]}
    ],
    "names": [ #实例名
      "robot_1",
      "robot_2"
    ]
  },
  {
    "kind": "Camera3D", #相机节点
    "funs": [
      {"f":"capture","errs":[]},
      {"f":"pose_recognize","errs":["failed"]}
    ],
    "names": [
      "camera_1",
      "camera_2"
    ]
  }
]\n
```

3.12 工作流-获取/设置

前端要去绘制一个场景的工作流程时，首先要去调用获取函数得到流程信息（中文翻译由前端完成）。修改完流程后调用设置函数去保存工作流程信息。

3.13 工作流-启动/停止

让整个场景工作起来，调用启动函数。需要突然终止则调用停止函数。