

20260118通过SW导出urdf

前言

虽然urdf文件是描述机构内部连杆和关节分布及关系的规范性语言，但是纯手写urdf文件是折磨的，特别是遇上复杂的机器人机构（如四足机器人）。现实中我们一般先进行SW建模，配合插件sw2urdf以及一定的规范自动生成urdf文件。

本文将以一个带伸缩臂和夹爪的四轮小车为例，讲解如何从SW导出urdf并实现小车在pybullet的仿真和控制，

一、下载sw2urdf插件

SW本身没有将结构导出为urdf的选项，需要我们下载插件sw2urdf将实现这一过程。

sw2urdf下载地址如下：

[https://wiki.ros.org/action/fullsearch/sw_urdf_exporter?
action=fullsearch&context=180&value=linkto:%22sw_urdf_exporter%22](https://wiki.ros.org/action/fullsearch/sw_urdf_exporter?action=fullsearch&context=180&value=linkto:%22sw_urdf_exporter%22)

SolidWorks to URDF Exporter

The SolidWorks to URDF exporter is a SolidWorks add-in that allows for the convenient export of SW Parts and Assemblies into a URDF file. The exporter will create a ROS-like package that contains a directory for meshes, textures and robots (urdf files). For single SolidWorks parts, the part exporter will pull the material properties and create a single link in the URDF. For assemblies, the exporter will build the links and create a tree based on the SW assembly hierarchy. The exporter can automatically determine the proper joint type, joint transforms, and axes.



If the above provided download fails to work on your system, please install by source before reporting an issue. If installing by source does work, please submit an [Update Installer Request](#)

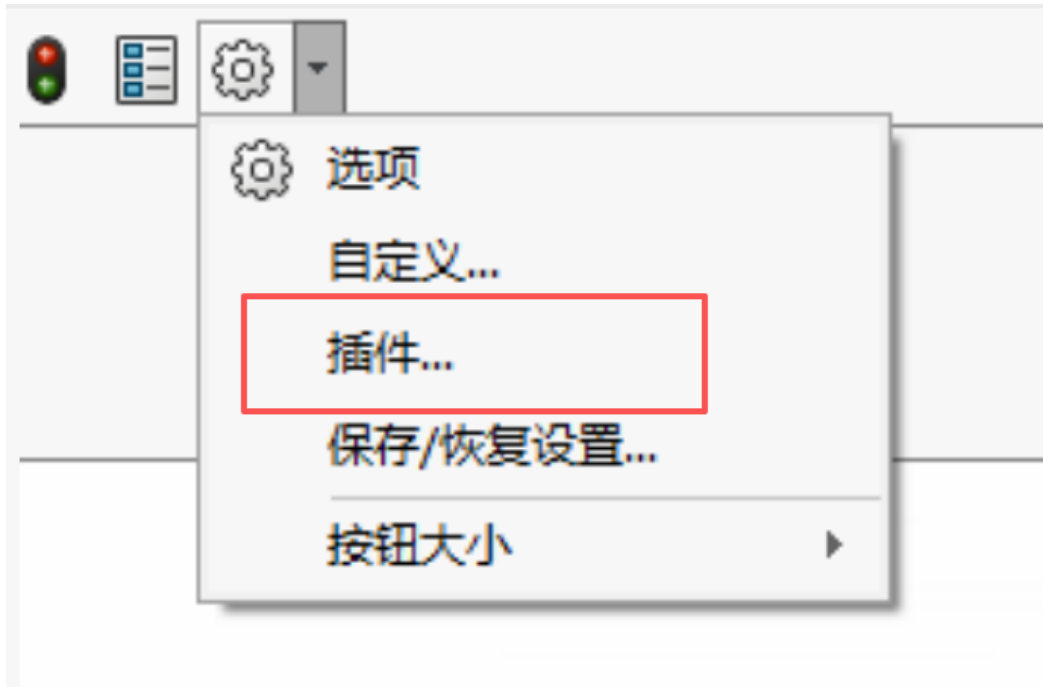
1. SolidWorks Version Compatibility

There is a known STL export bug with SolidWorks 2018 that exists up to Service Pack 4 that renders this add-in unusable. If you are using 2018, please update to service pack 5 or use SolidWorks 2019 or later. 2017 and below may also work

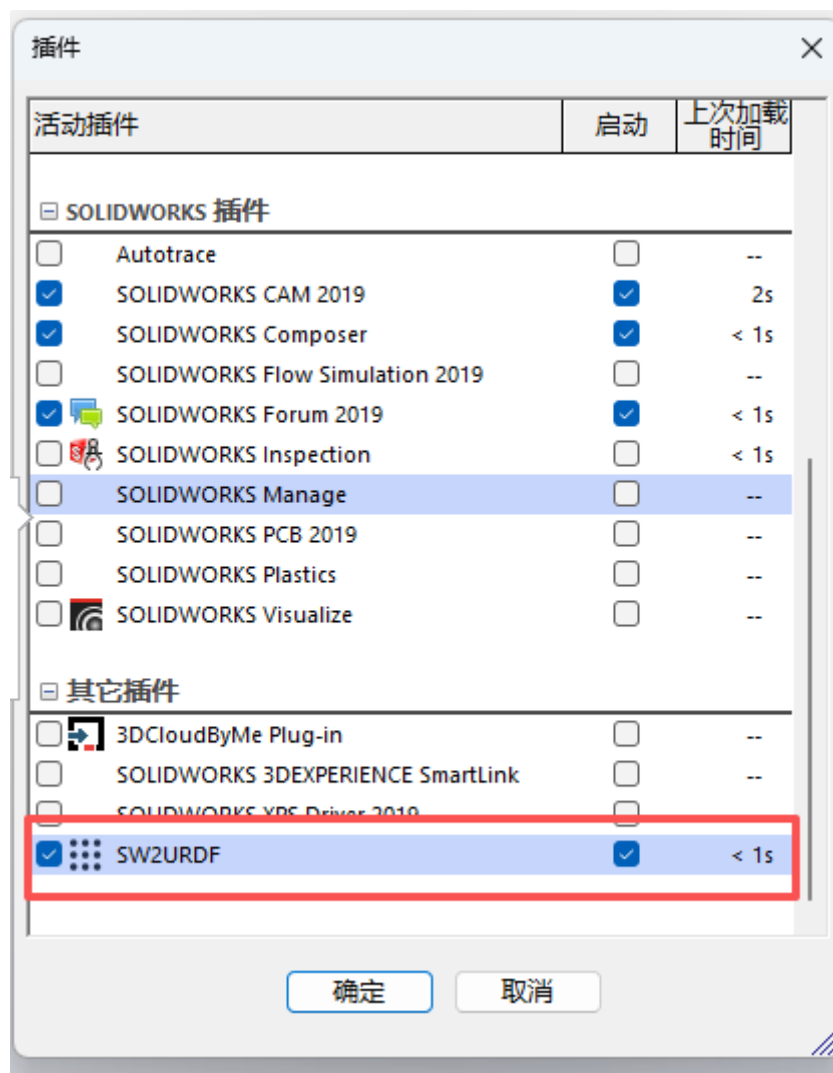
进入页面后点击“**Download Installer**”的蓝色按钮即可下载。



下载安装包后随引导安装即可。安装完毕后，在sw界面点开齿轮图标旁的小倒三角，点击“插件”，



打开插件界面后下滑找到SW2URDF代表安装成功。

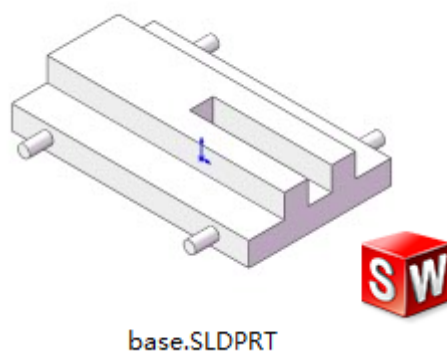


二、导出模型前的准备工作

为了能顺利导出urdf模型，我们需要按照一定的规范将我们的四轮小车模型组织起来。

2.1.建模小车各个零件

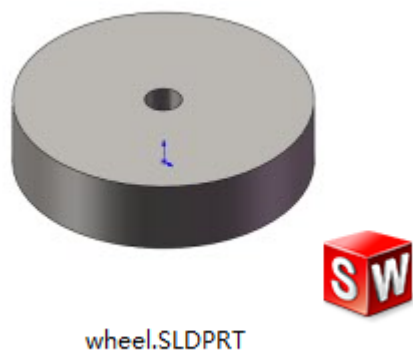
小车底盘base:



这个底盘比较简单，两侧各引出2个对称的定位柱便于后续装配过程中轮子的定位和配合。

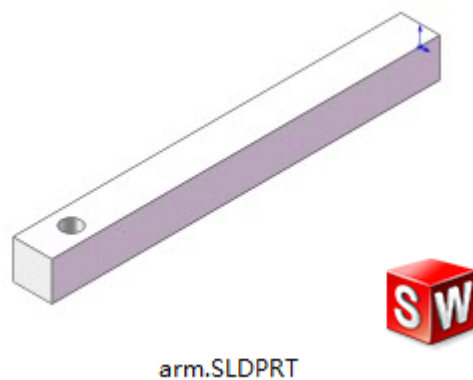
中间则是挖了个槽用来安装伸缩臂，虽然这不是必要的，但是方便后续能直观观察伸缩臂的运动，所以这里也建模出来了。

小车轮子wheel



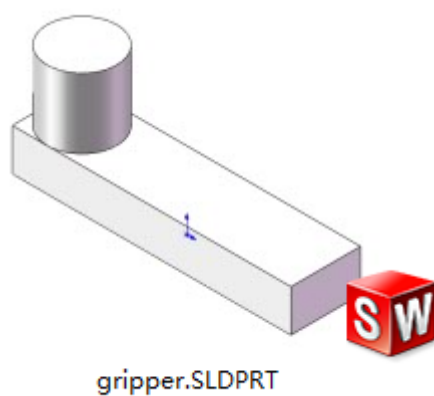
小车轮子用于体现urdf中continuous的效果，就是一个中空的圆盘，虽然我们实际用到4个轮子，这里实际建模一个就好。

伸缩臂arm



伸缩臂用于体现urdf中prismatic的效果就是一个长方体，一端打了圆孔方便夹爪的定位。

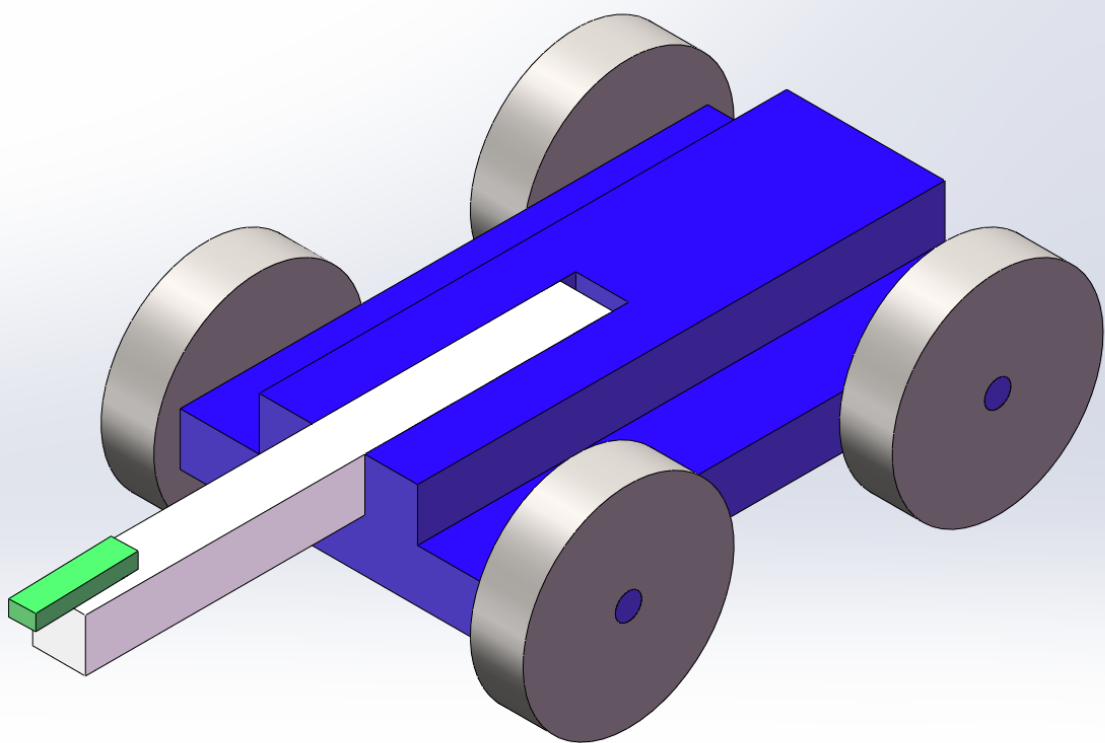
夹爪gripper



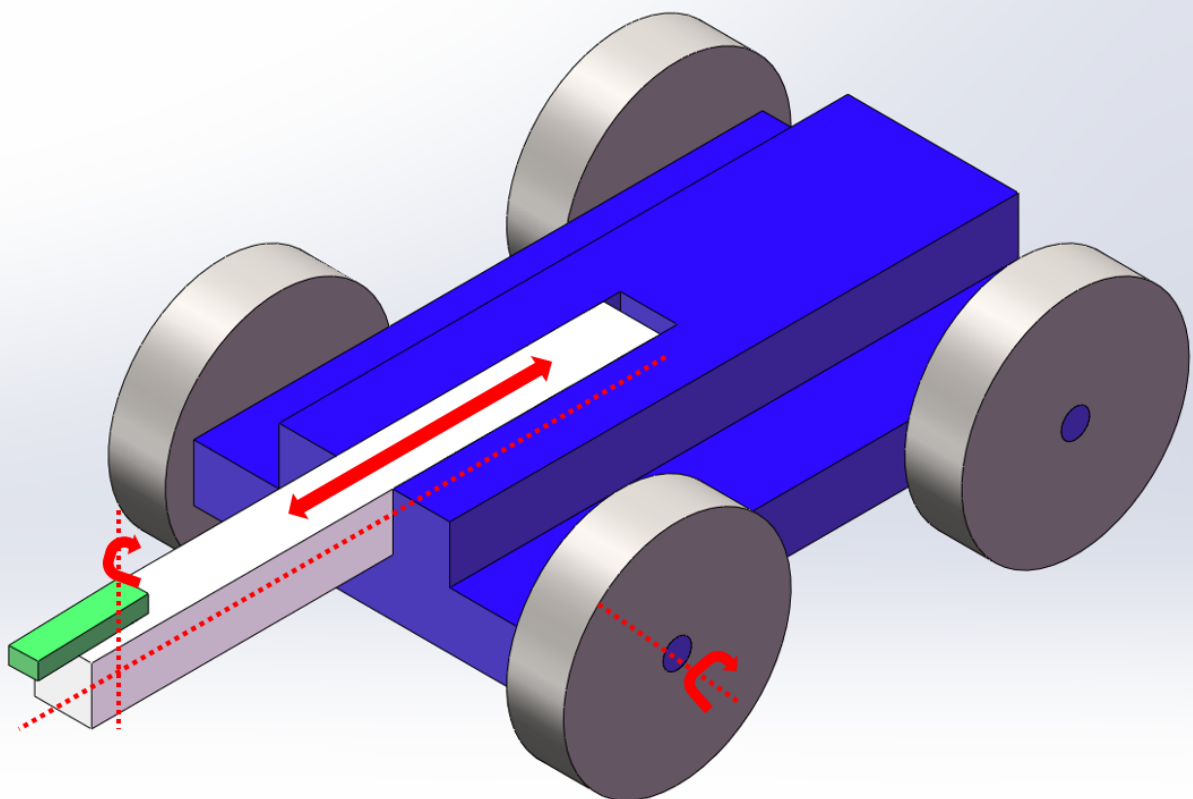
这个夹爪略显抽象，不过建模这个夹爪主要是用于体现urdf中revolute，也还算凑合。一端引出的圆柱用于与伸缩臂的圆孔配合，实现定位。

2.2 小车整体装配

按照零件以及我们设想的装配，最终的装配结果如下



各构件的运动关系如下图表示



三、为装配体标注参考坐标系和参考轴

3.1.为什么要为装配体标注参考坐标系和参考轴？

观察urdf中定义关节的代码段

代码块

```
1 <joint name="front_left_wheel_joint" type="continuous">
2   <parent link="base_link"/>
3   <child link="front_left_wheel"/>
4   <origin xyz="0.2 0.15 0" rpy="1.5708 0 0"/>
5   <axis xyz="0 0 1"/>
6 </joint>
```

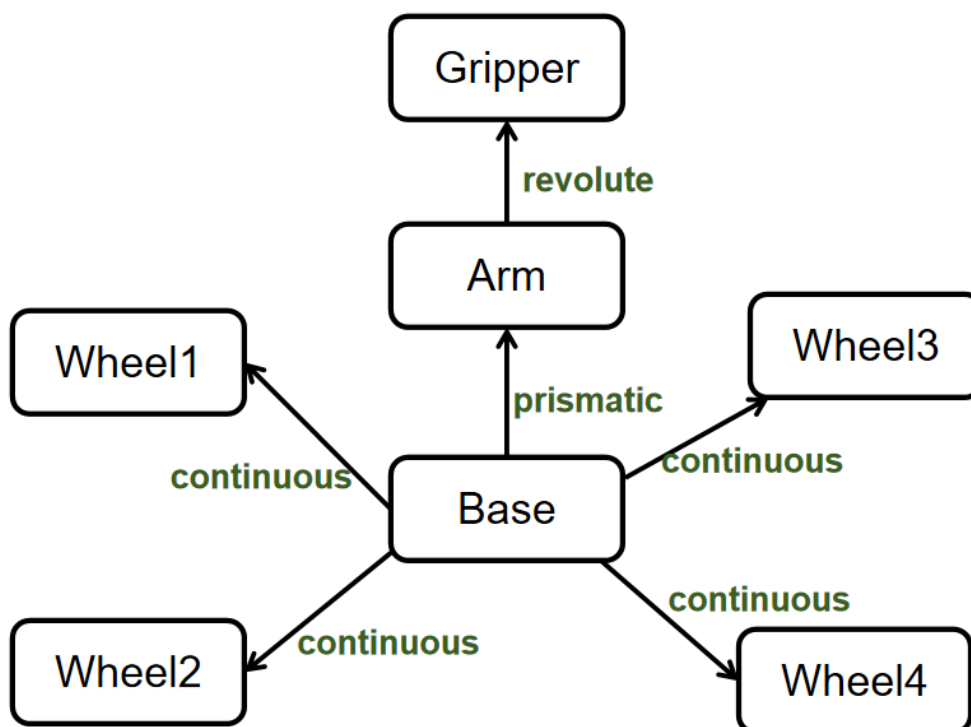
`<origin xyz="0.2 0.15 0" rpy="1.5708 0 0"/>` 定义了子连杆的坐标系原点（通常也是关节的坐标）

`<axis xyz="0 0 1"/>` 则定义该关节是绕着（沿着）哪根轴运动的。

在装配体中标注参考坐标系和参考轴实际上就是将上述urdf代码可视化手写了一遍，这是必要的。

3.2.需要标注多少个参考系和参考轴？

这个是根据机器人的关节总数决定的，有 n 个关节，就需要建立 n 个参考坐标系和 n 个坐标轴，外加一个base的参考坐标系。



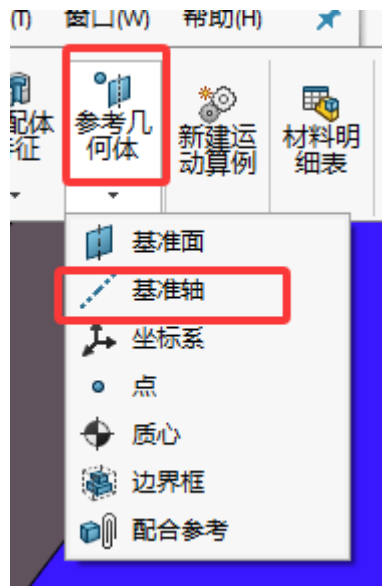
参考我们小车的机器人运动学树，其中的每一个箭头就代表一个关节，我们总共需要建立6个参考坐标和参考轴，外加一个Base的参考坐标系。

3.3.如何标注参考系和参考轴

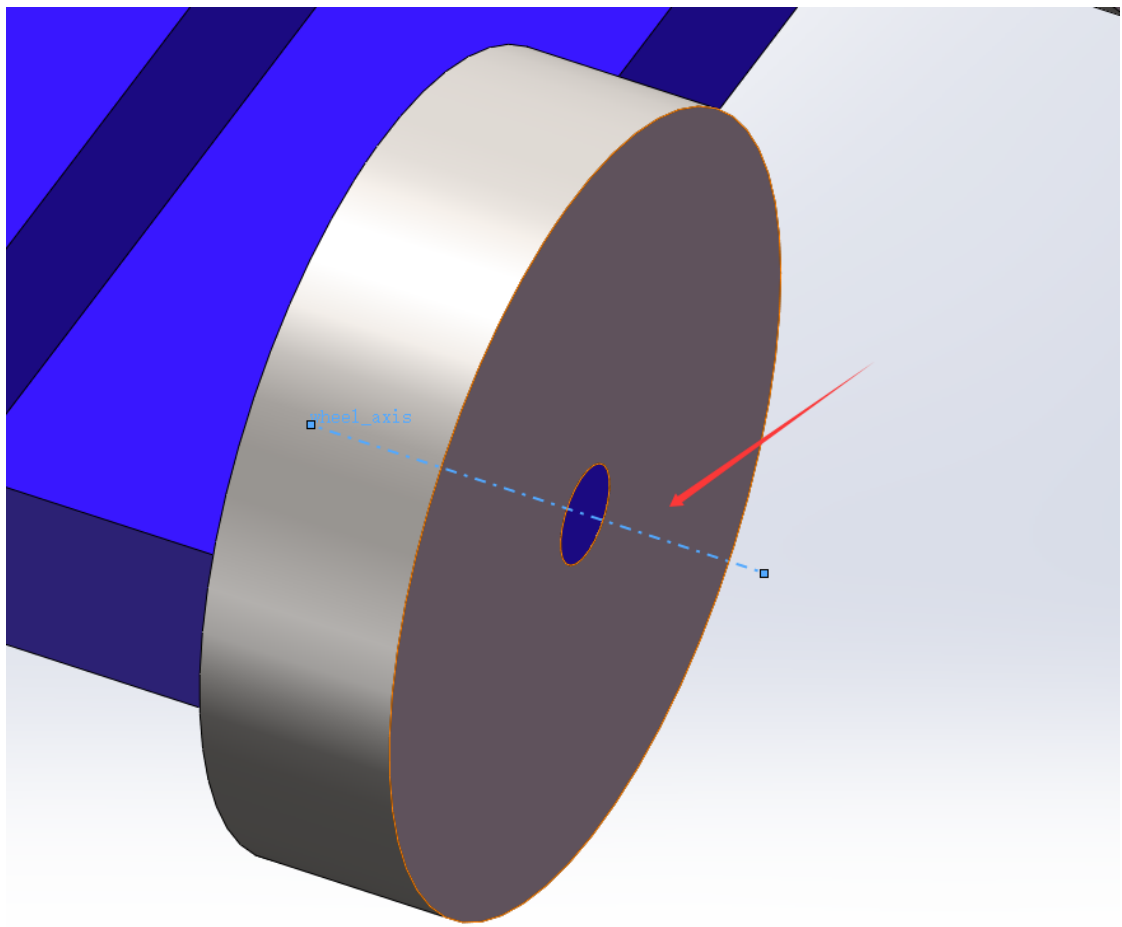
建立参考坐标系我们可以遵循以下步骤：

①建立参考轴

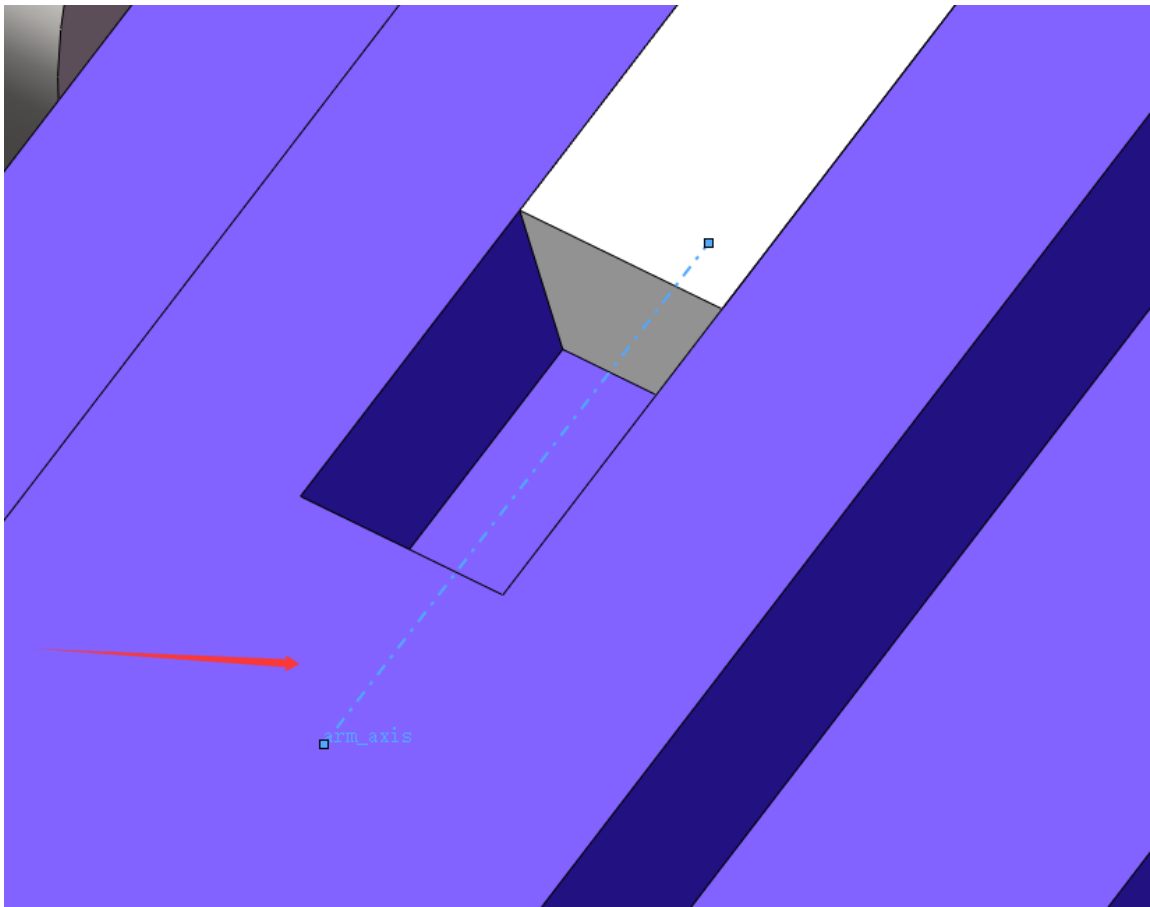
这一步比较简单，对于旋转运动（如小车轮子），绕着哪个轴旋转，这就是我们的参考轴；对于平移运动（如伸缩臂），沿着哪个轴移动，这就是我们的参考轴



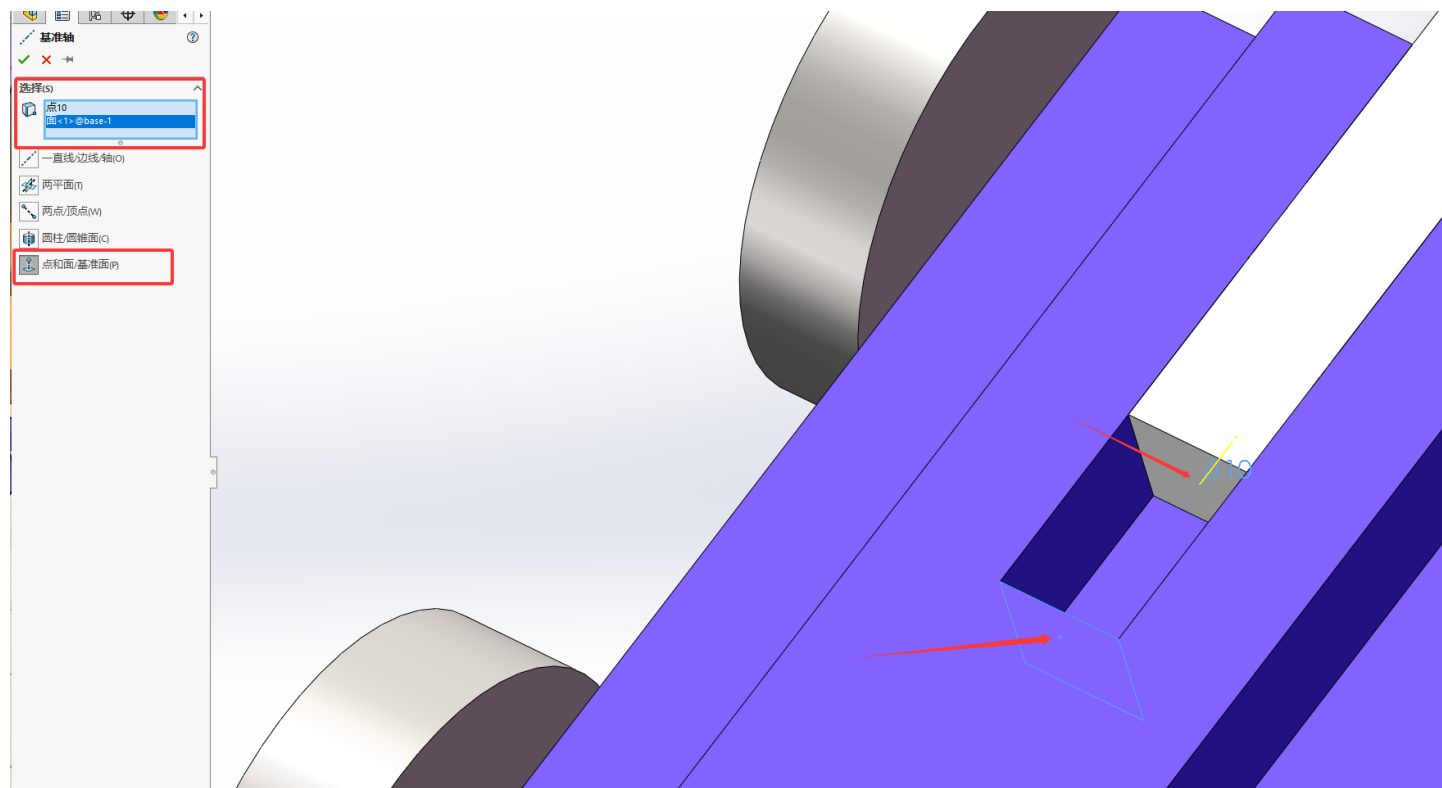
在sw “参考几何体” 选项卡选择 “基准轴” 即可创建参考轴，按照引导即可在对应位置创建参考轴
车轮的参考轴建立如下



伸缩臂建立基准轴如下



注：建立伸缩臂的参考轴稍微麻烦点，推荐使用点和基准面的建立方式，在伸缩臂中心取一个点，然后再取合适的平面，即可建立一条过伸缩臂中心的参考轴



②取点

取点方法是“参考几何体”——“点”



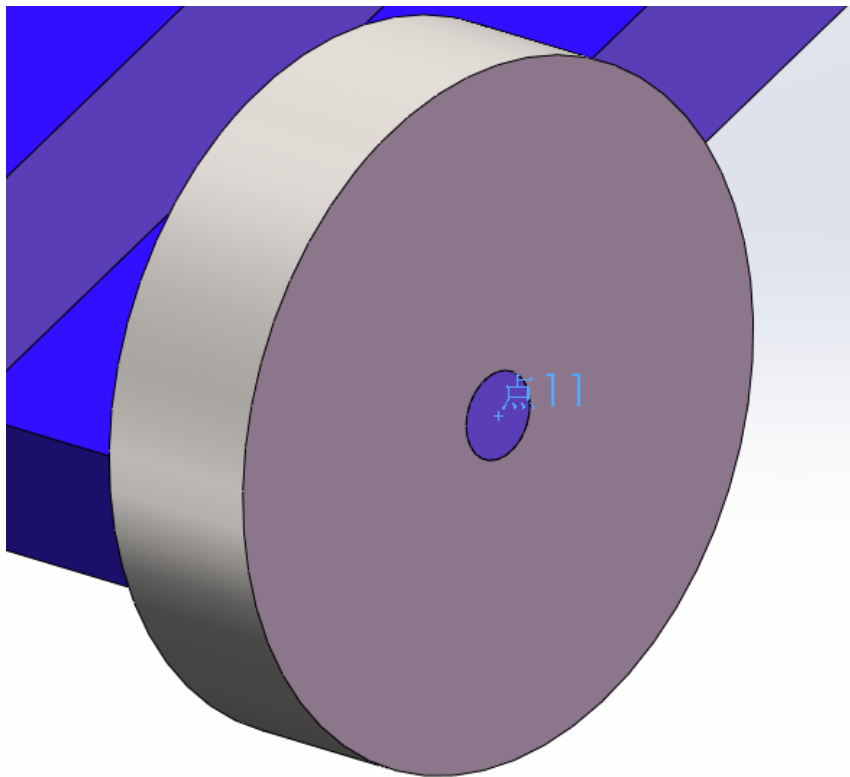
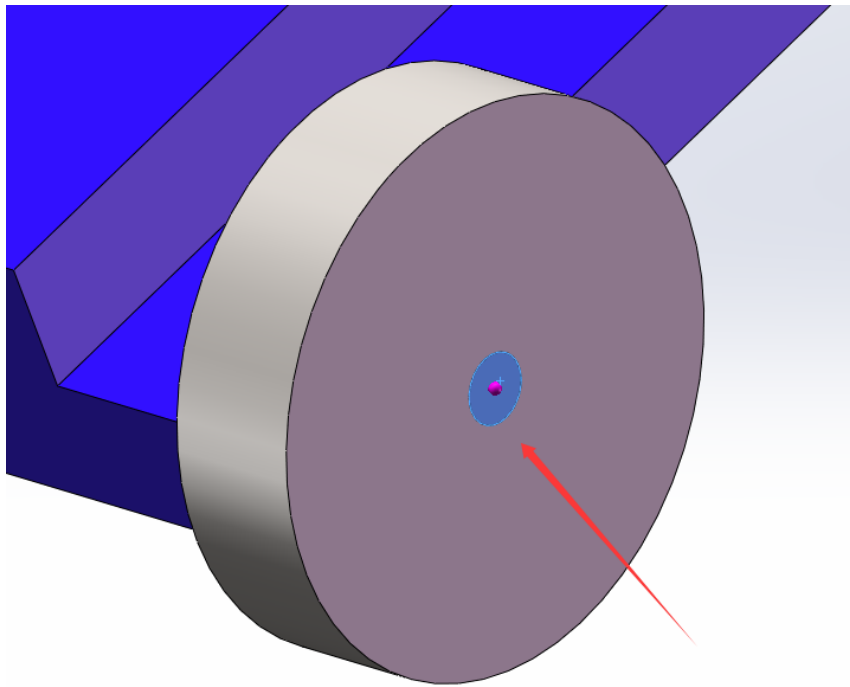
a.取点是为了确保参考坐标系处于正确的位置，通常这个点位于父连杆和子连杆相连接的关节的中心上

b.这个点通常点在父连杆上

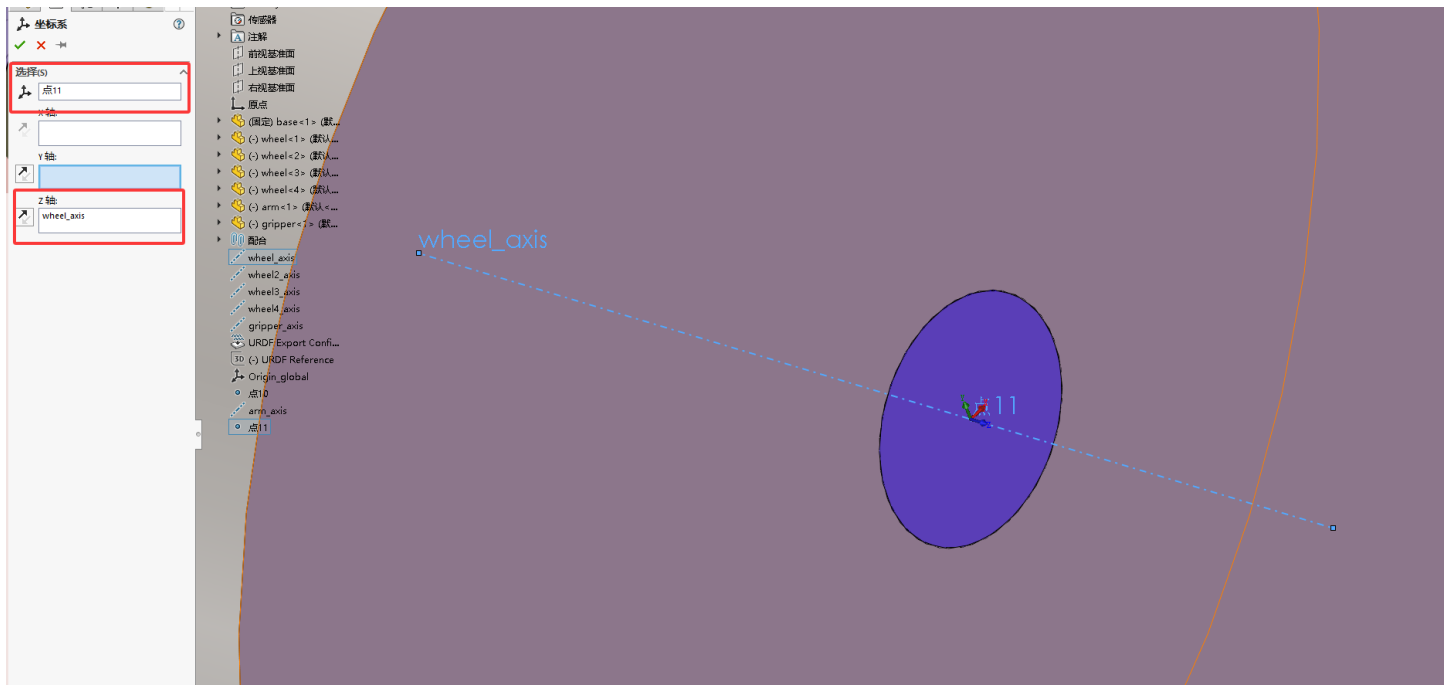
c.坐标系的Z与实际旋转轴和

比如：

对于车轮wheel，其与底盘相连接的关节位于图中箭头的指示位置，我们可以在这个位置，在父连杆上取一个点。



随后在“参考几何体”——“坐标系”建立参考坐标系



坐标系的取点选择我们刚取好的点11，Z轴选择之前建立的参考轴wheel_axis，x轴和y轴的方向符合右手定则即可(在SW上保持默认)。

对于伸缩臂arm



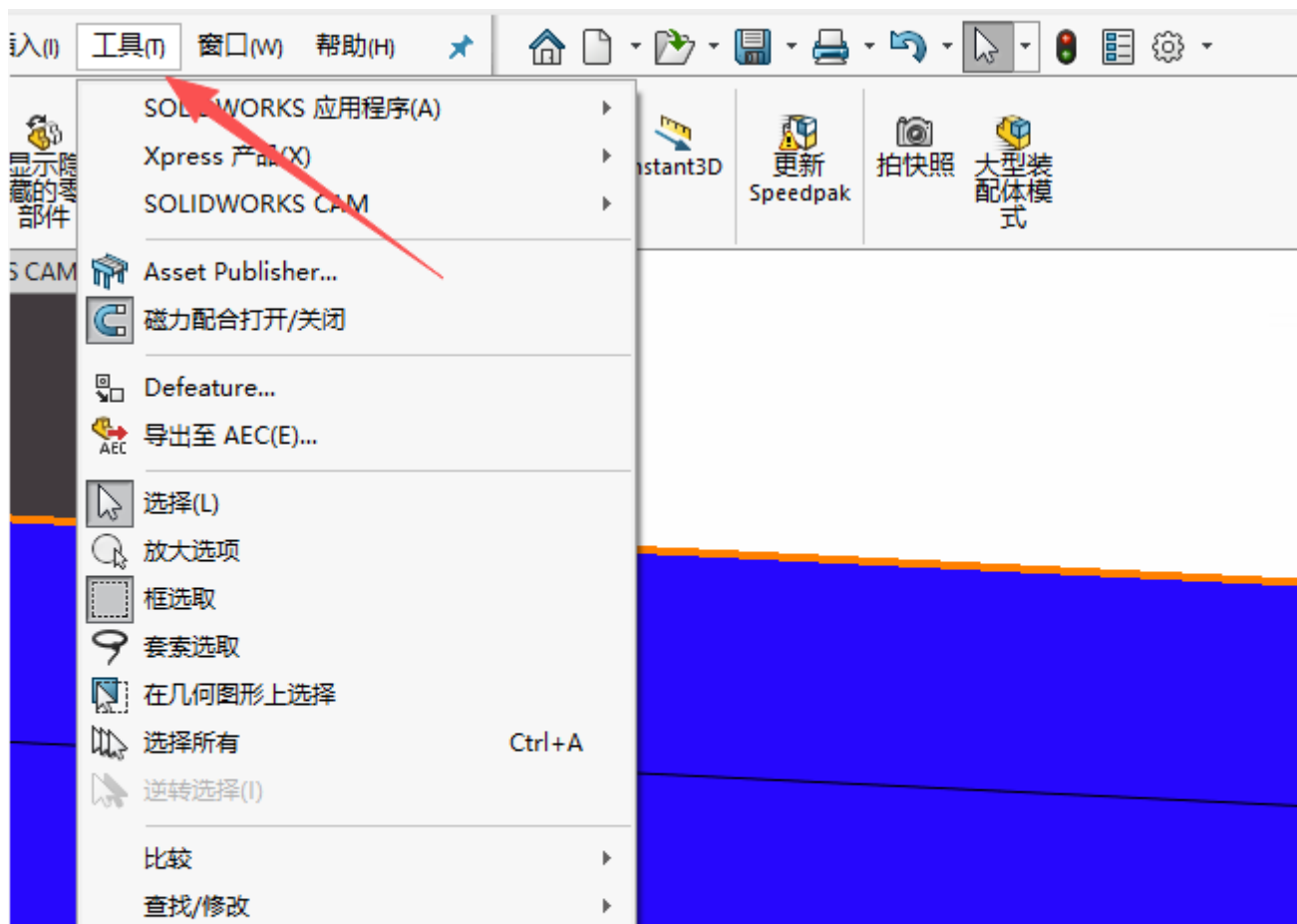
随后依次建立各个参考轴和参考坐标系如下，7个参考坐标系和6个参考轴（请一定要用英文命名好!!!）

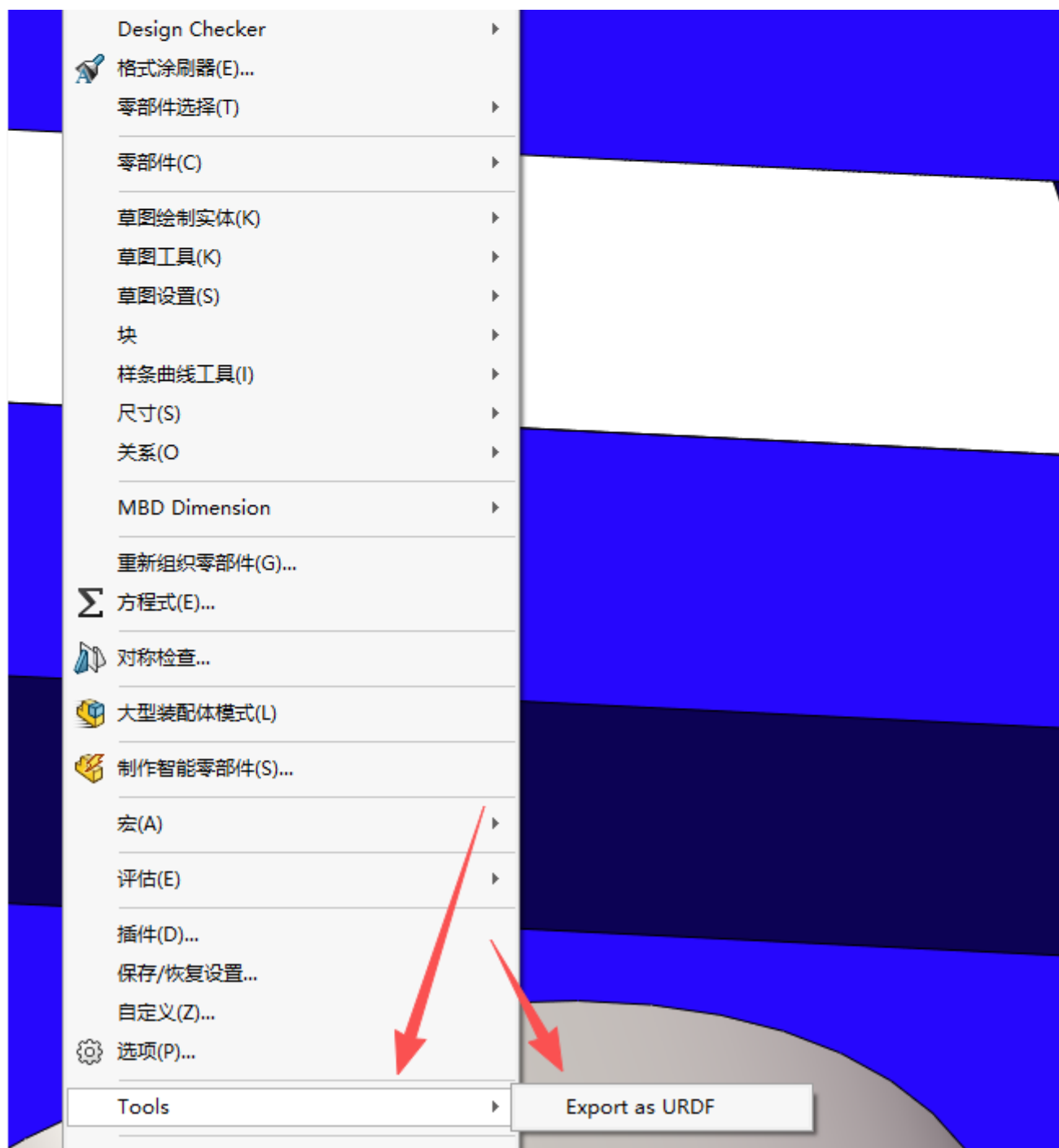


四、配置导出urdf选项卡

4.1打开导出urdf文件配置界面

在SW界面**工具**——**Tools**——**Export as URDF**打开URDF导出配置界面





打开的界面如下

Configure and Organize Links

base_link 1

Global Origin Coordinate System

Automatically Generate 2

3

0 4

Load Configuration...

Preview and Export...

....base_link

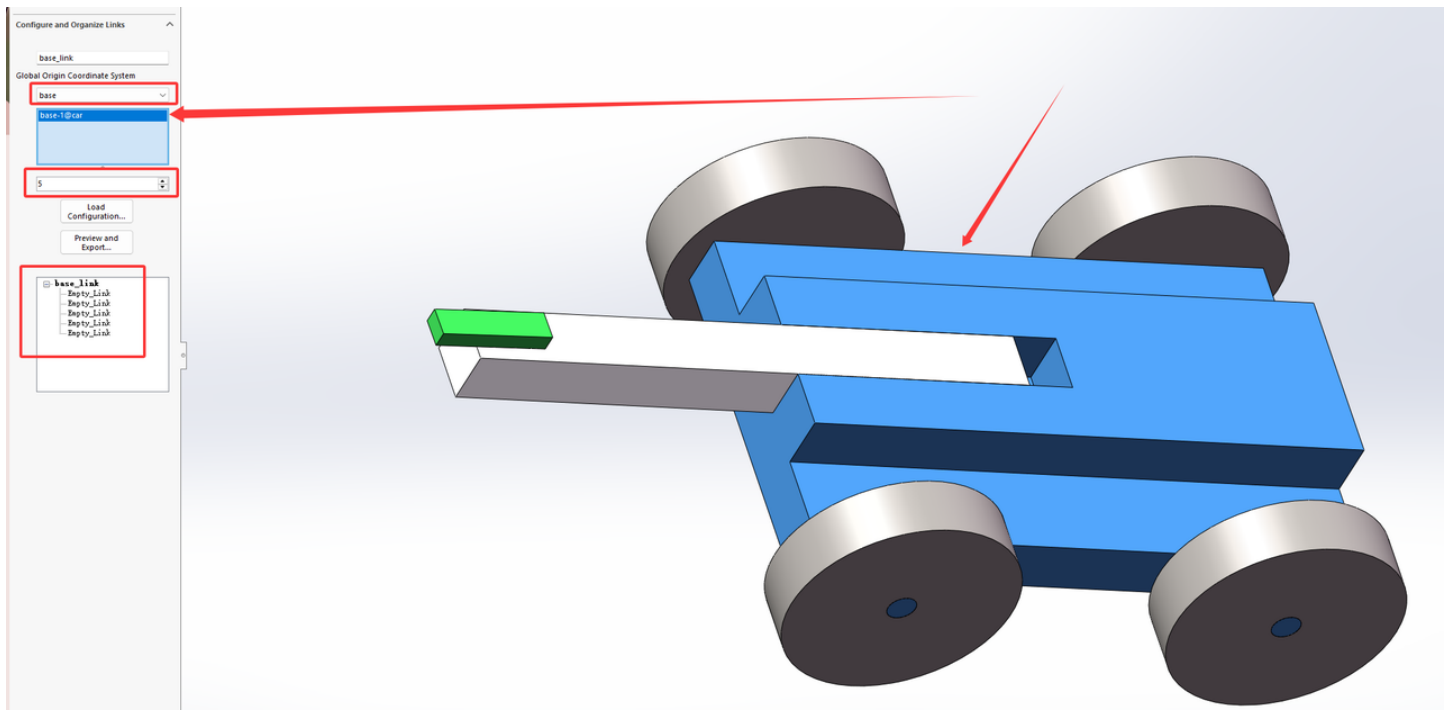
第1个选项框表示配置当前连杆的名字

第2个选项框表示配置当前连杆的参考坐标系

第3个选项框表示配置属于当前连杆的实际建模

第4个选项框表示配置该连杆下有多少个子连杆

对于我们的小车底盘base，其下有5个子连杆，我们可以这样配置。



可以发现，左边的分支树多了5个Empty_Link，我们点击任意一个Empty_Link，得到界面如下：

The image shows a software interface titled "Configure and Organize Links". It contains several configuration fields and buttons, with red numbers 1 through 7 pointing to specific elements:

- 1** points to the "base_link" dropdown menu, which currently shows "Empty_Link".
- 2** points to the "Joint Name" text input field.
- 3** points to the "Reference Coordinate System" dropdown menu, which shows "Automatically Generate".
- 4** points to the "Reference Axis" dropdown menu, which also shows "Automatically Generate".
- 5** points to the "Joint Type" dropdown menu, which shows "Automatically Detect".
- 6** points to a large blue rectangular area representing a 3D model of a link.
- 7** points to a numeric input field showing the value "0", which represents the number of child links.

Below the numbered fields are two buttons: "Load Configuration..." and "Preview and Export...". At the bottom, there is a tree view showing a hierarchy starting with "base_link" and containing several "Empty_Link" entries.

第1个选项框表示配置当前连杆的名字

第2个选项框表示配置当前连杆和父连杆之间关节的名字

第3个选项框表示配置当前连杆的参考坐标系

第4个选项框表示配置当前连杆运动的参考轴

第5个选项框表示配置当前连杆的关节类型

第6个选项框表示配置当前连杆的实际建模

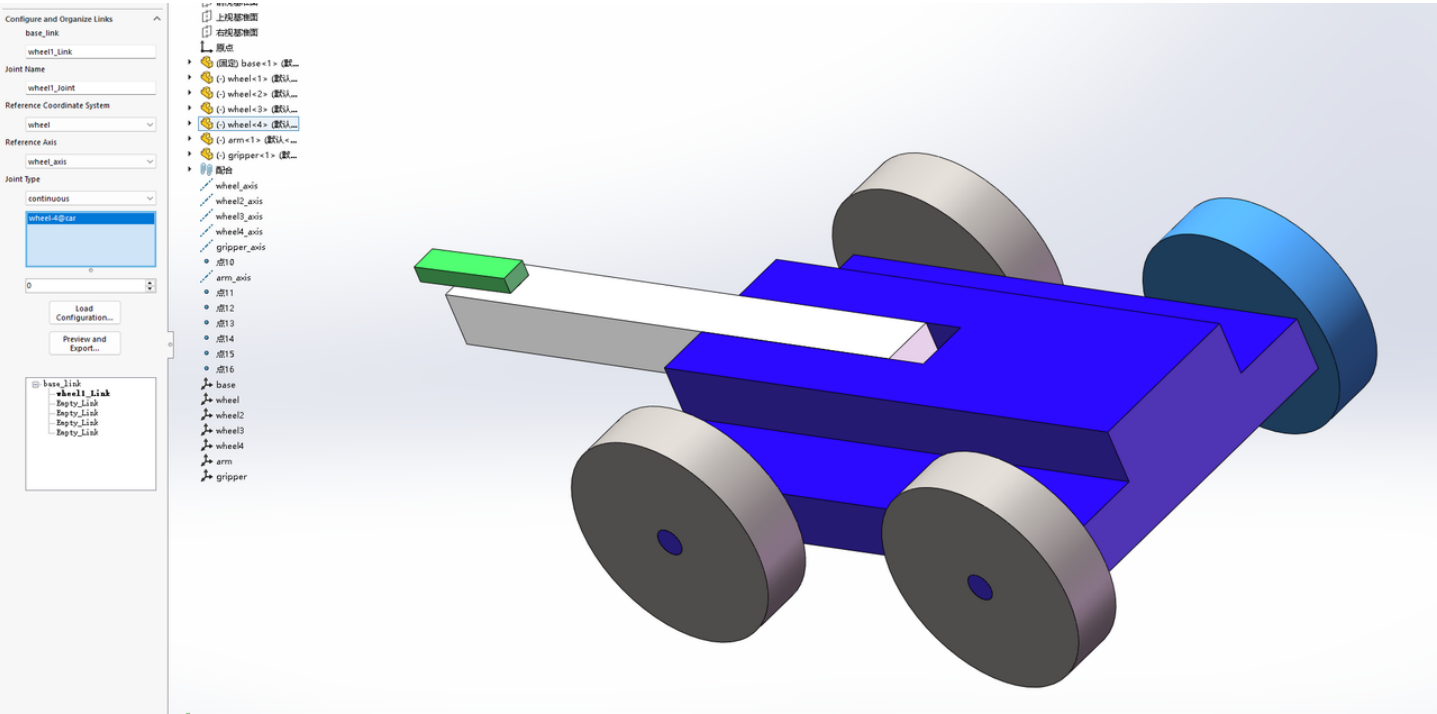
第7个选项框表示配置当前连杆下有多少个子连杆。

需要特别注意的是**第5个选项框表示配置当前连杆的关节类型**

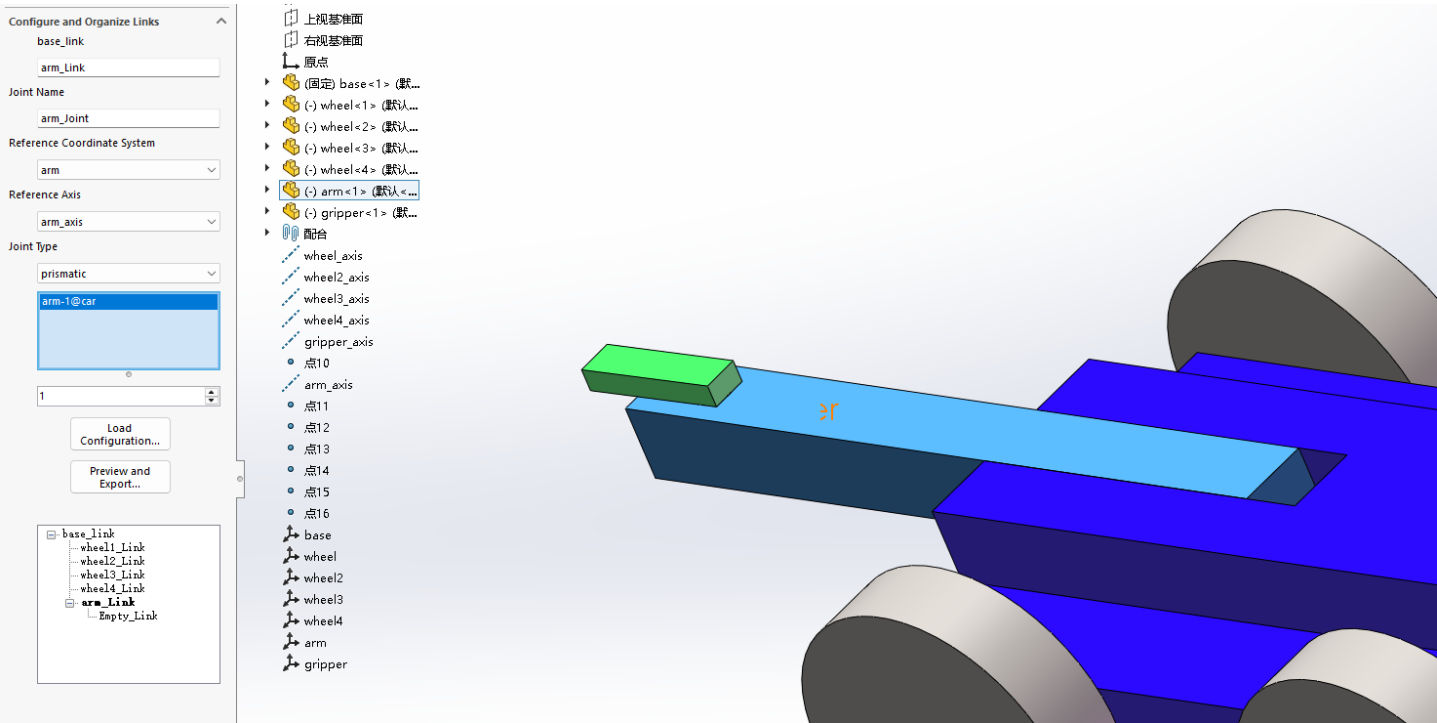
选择什么样的关节类型可以根据以下表格确定：

关节类型	运动性质	运动限制	关键参数	典型应用场景
revolute	旋转	有（角度范围）	必须定义 <limit> (upper/lower)	机械臂的肘部、膝盖、手指关节
continuous	旋转	无（无限循环）	不需要 <limit>	机器人轮子、螺旋桨、传送带
prismatic	平移	有（线性距离）	必须定义 <limit> (upper/lower)	电动推杆、伸缩腿、滑轨、电梯

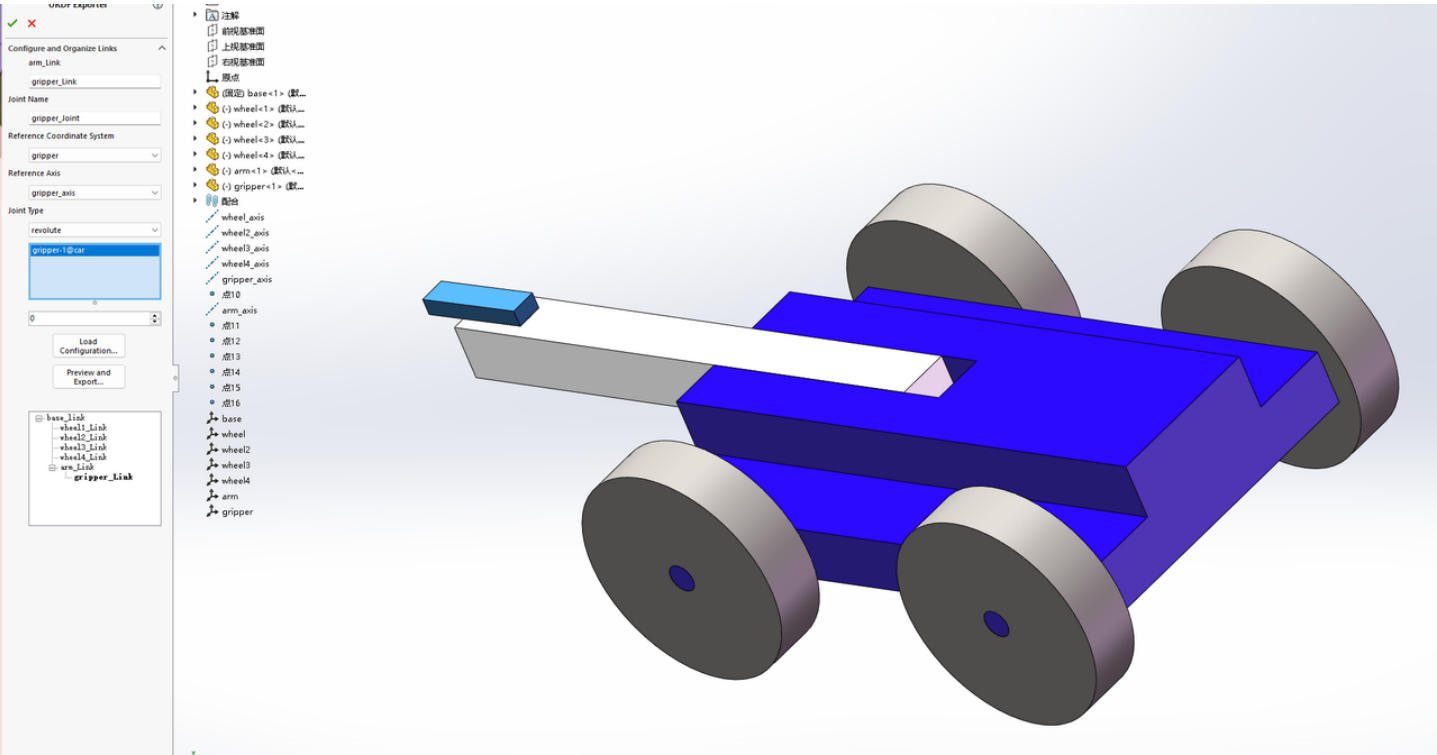
比如对于wheel我们这样配置，由于我们已经提前配置好了参考坐标系wheel和参考坐标轴wheel_axis并命名好，这一步比较轻松，注意选择wheel对应的关节类型是continuous就好



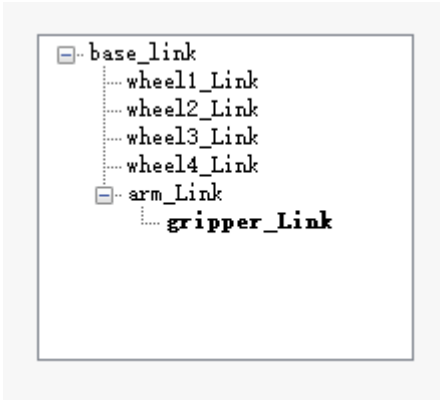
arm的配置如下



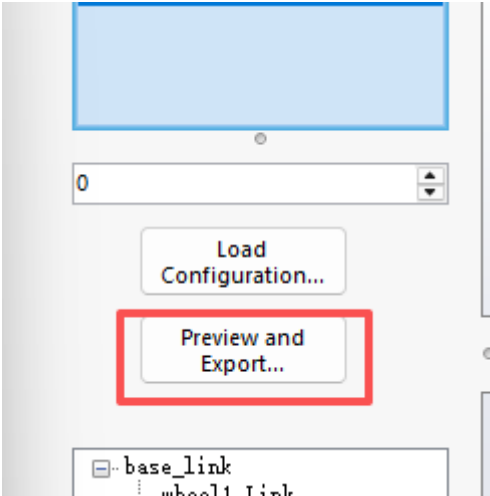
gripper的配置如下



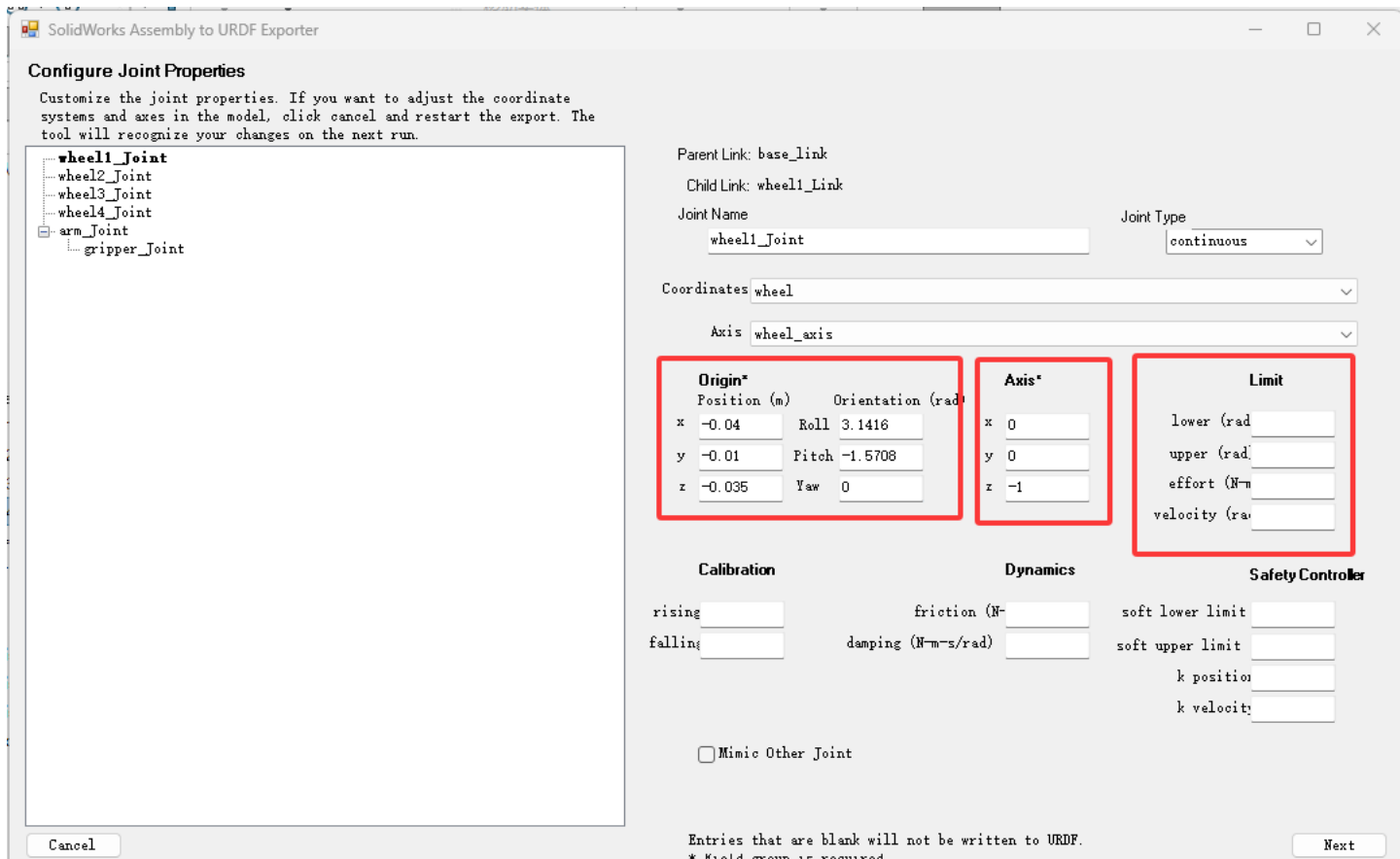
最后的urdf树如下



这个时候就可以点击“Preview and Export”进行导出



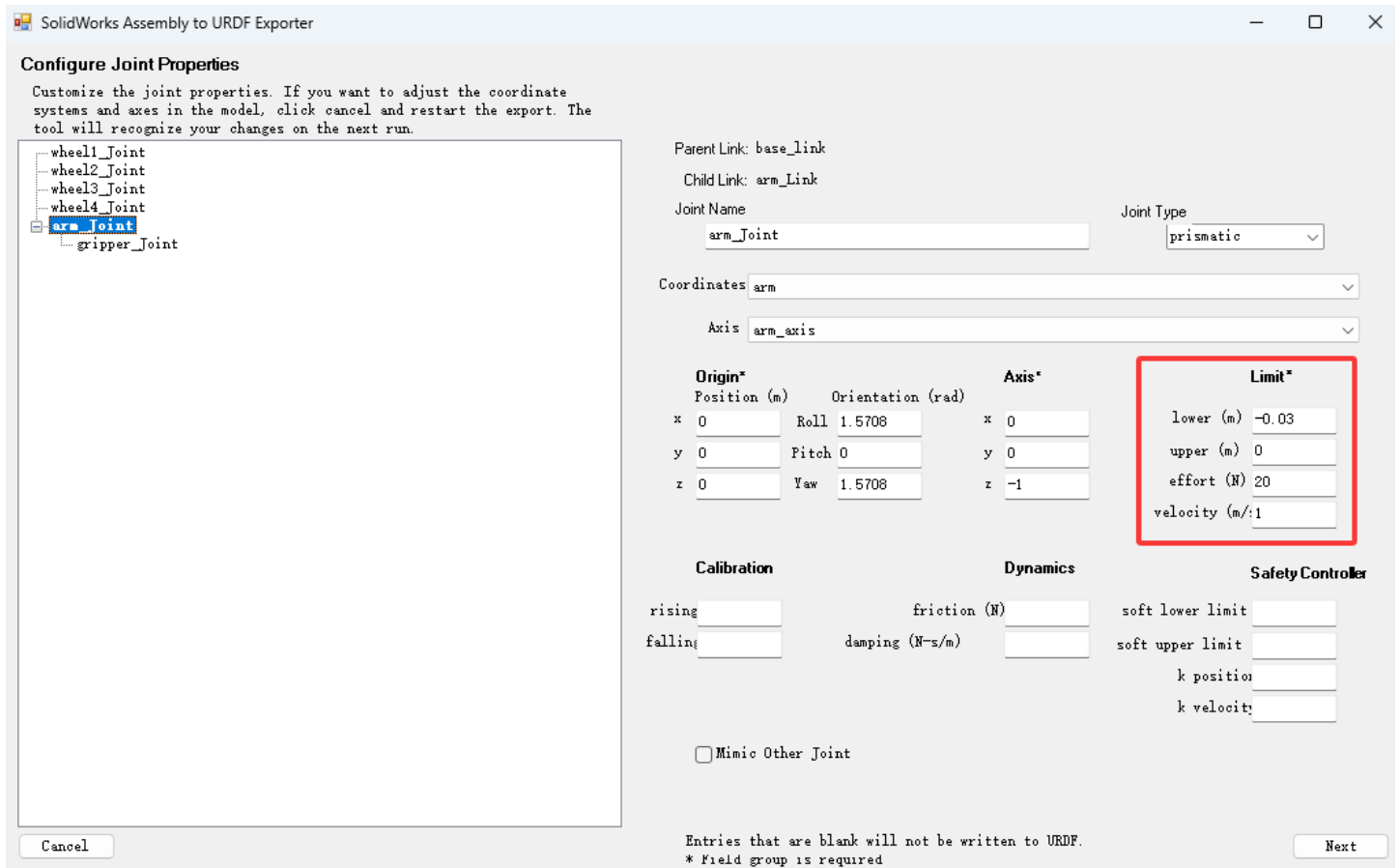
打开的界面如下：



首先进入的是关节配置界面，点开左边任意joint查看，发现sw2urdf已经自动帮我们配好了Origin和Axis，这些数据都是基于我之前配置好的参考坐标和参考轴生成的。

值得注意的是，arm的关节类型prismatic是和gripper的关节类型是revolute，这两个类型的Limit需要我们自己配置，点开arm_Joint和gripper_Joint进行配置

arm_Joint配置如下



lower（位置最小值），这里我配置-0.03m

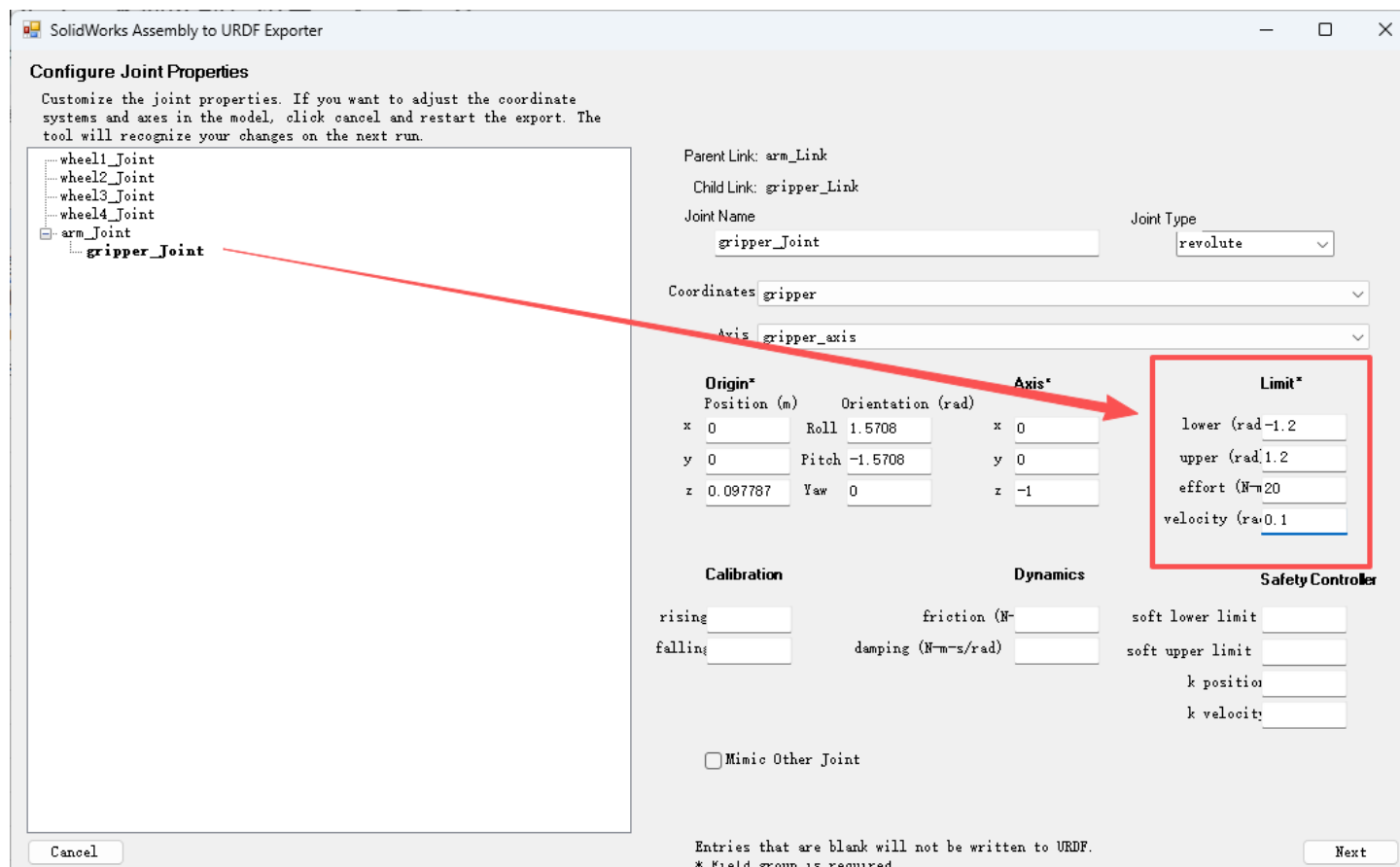
upper（位置最大值）,这里我配置0.00m

effort（最大推力），这里我配置20N

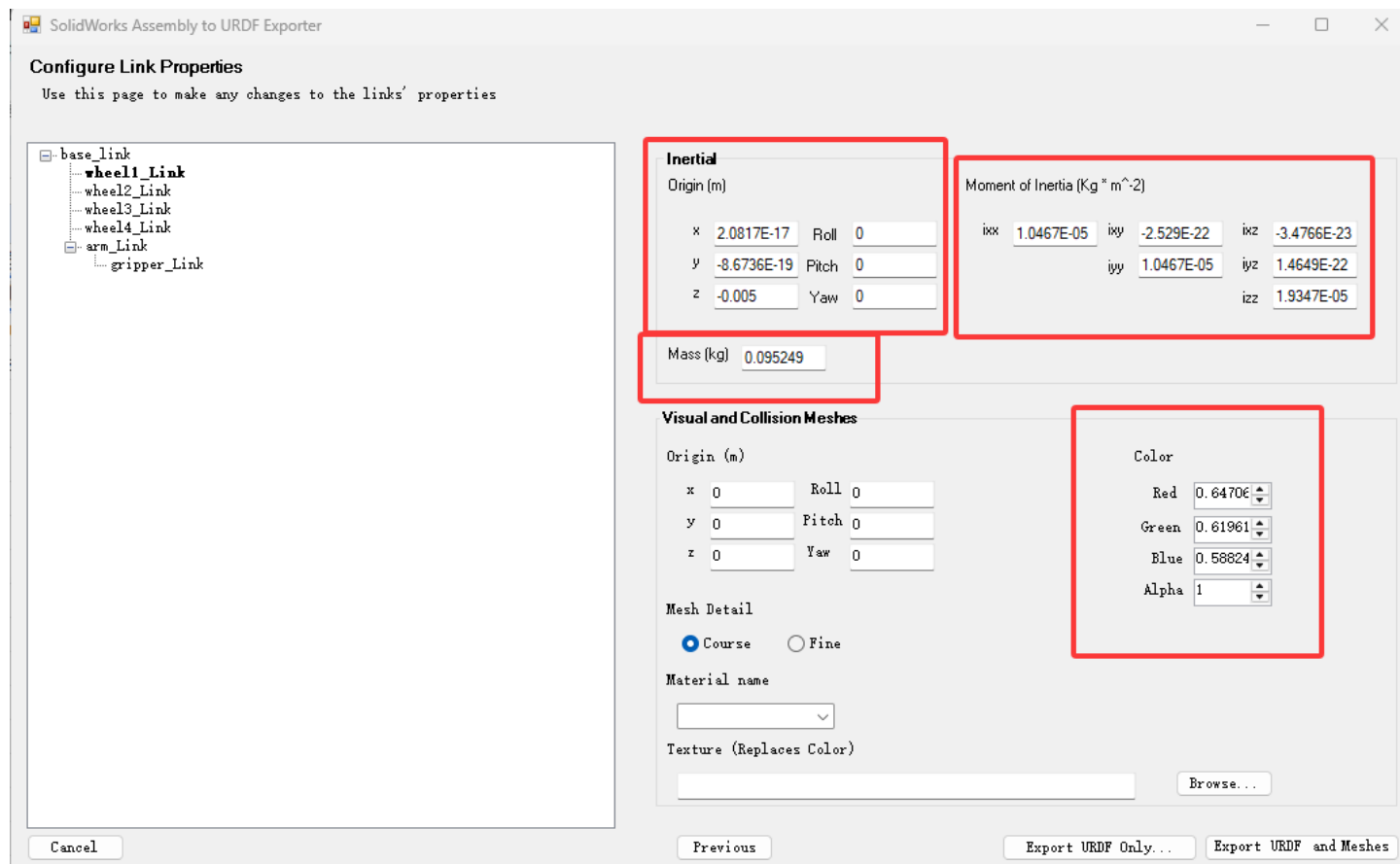
velocity(最大速度)，这里我配置1m/s

大家根据自己的实际情况配置即可

gripper_Joint的配置如下

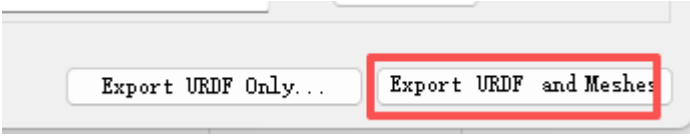


随后点击右下角的**Next**，来到连杆配置界面

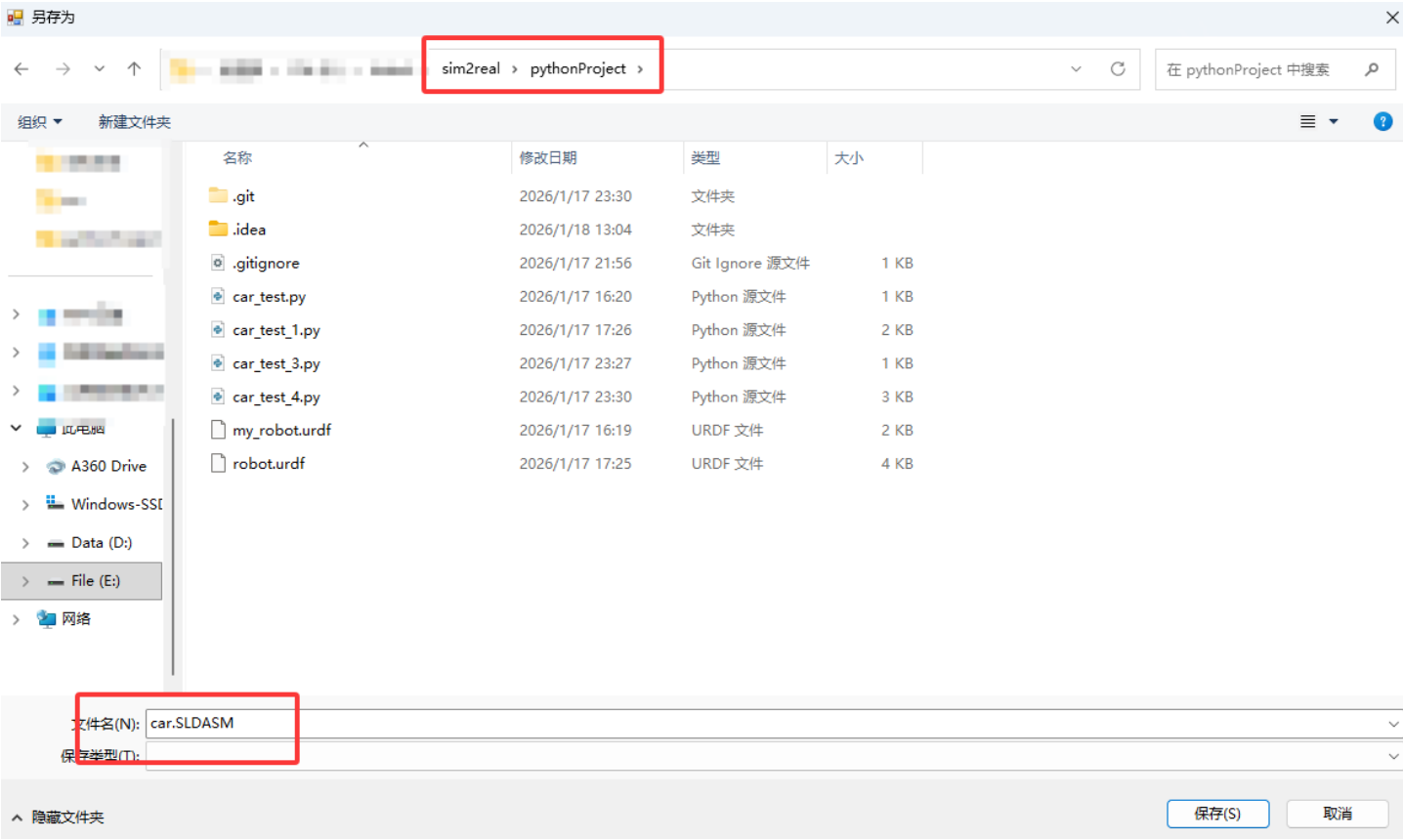


sw2urdf已经自动根据我们连杆的实际建模和实际使用材料（在SW中进行材料配置）帮我们计算好了质量，转动惯量等动力学参数，我们保持默认即可，读者也可以根据自己的需要进行修正。

我们选择右下角**Export URDF and Meshes**保存文件










笔者将文件命名为car.SLDASM保存在了调试pybullet的工程目录下



五、在pybullet打开导出好的小车urdf文件

导出的car.SLDASM文件内容如下，urdf文件夹就保存着我们需要的.urdf文件，meshes文件夹下保存着实际建模的.stl文件，也很重要。

config	2026/1/18 14:17	文件夹	
launch	2026/1/18 14:17	文件夹	
meshes	2026/1/18 14:17	文件夹	
textures	2026/1/18 14:17	文件夹	
urdf	2026/1/18 14:17	文件夹	
CMakeLists.txt	2026/1/18 14:17	文本文档	1 KB
export.log	2026/1/18 14:17	文本文档	1,229 KB
package.xml	2026/1/18 14:17	Microsoft Edge ...	1 KB
car.SLDASM.csv	2026/1/18 14:17	Microsoft Excel ...	4 KB
car.SLDASM.urdf	2026/1/18 14:17	URDF 文件	9 KB

 arm_Link.STL	2026/1/18 14:17	BambuStudio	61 KB
 base_link.STL	2026/1/18 14:17	BambuStudio	61 KB
 gripper_Link.STL	2026/1/18 14:17	BambuStudio	61 KB
 wheel1_Link.STL	2026/1/18 14:17	BambuStudio	61 KB
 wheel2_Link.STL	2026/1/18 14:17	BambuStudio	61 KB
 wheel3_Link.STL	2026/1/18 14:17	BambuStudio	61 KB
 wheel4_Link.STL	2026/1/18 14:17	BambuStudio	61 KB

在工程文件下编写测试代码**cat_test.py**，内容如下：

请保证 **cat_test.py** 和 **car.SLDASM** 在同一目录下

代码块

```

1  import pybullet as p
2  import os
3  import time
4  import pybullet_data
5
6  # 1. 初始化仿真环境
7  p.connect(p.GUI)
8  p.setAdditionalSearchPath(pybullet_data.getDataPath())
9  p.setGravity(0, 0, -9.8)
10
11 # 加载地面
12 p.loadURDF("plane.urdf")
13
14 # 2. 定位并加载你的 URDF 模型
15 # 请确保此脚本与 model 文件夹在同一目录下
16 urdf_path = os.path.join(os.getcwd(), "car.SLDASM/urdf/car.SLDASM.urdf")
17
18 # 加载机器人，并稍微抬高初始位置防止掉入地面
19 robot_id = p.loadURDF(urdf_path, [0, 0, 0.1], useFixedBase=False)
20
21 # 3. 创建控制滑块 (Debug Parameters)
22 # 参数名称，最小值，最大值，初始值
23 # 控制前进/后退的速度 (-15 到 15)
24 vel_slider = p.addUserDebugParameter("Car Velocity", -15, 15, 0)
25
26 # 控制伸缩杆位置 (参考 URDF limit: -0.03 到 0.00)
27 arm_slider = p.addUserDebugParameter("Arm Extension", -0.03, 0.00, 0)
28
29 # 控制爪子开合 (参考 URDF limit: -1.2 到 1.2)
30 gripper_slider = p.addUserDebugParameter("Gripper Rotation", -1.2, 1.2, 0)
31

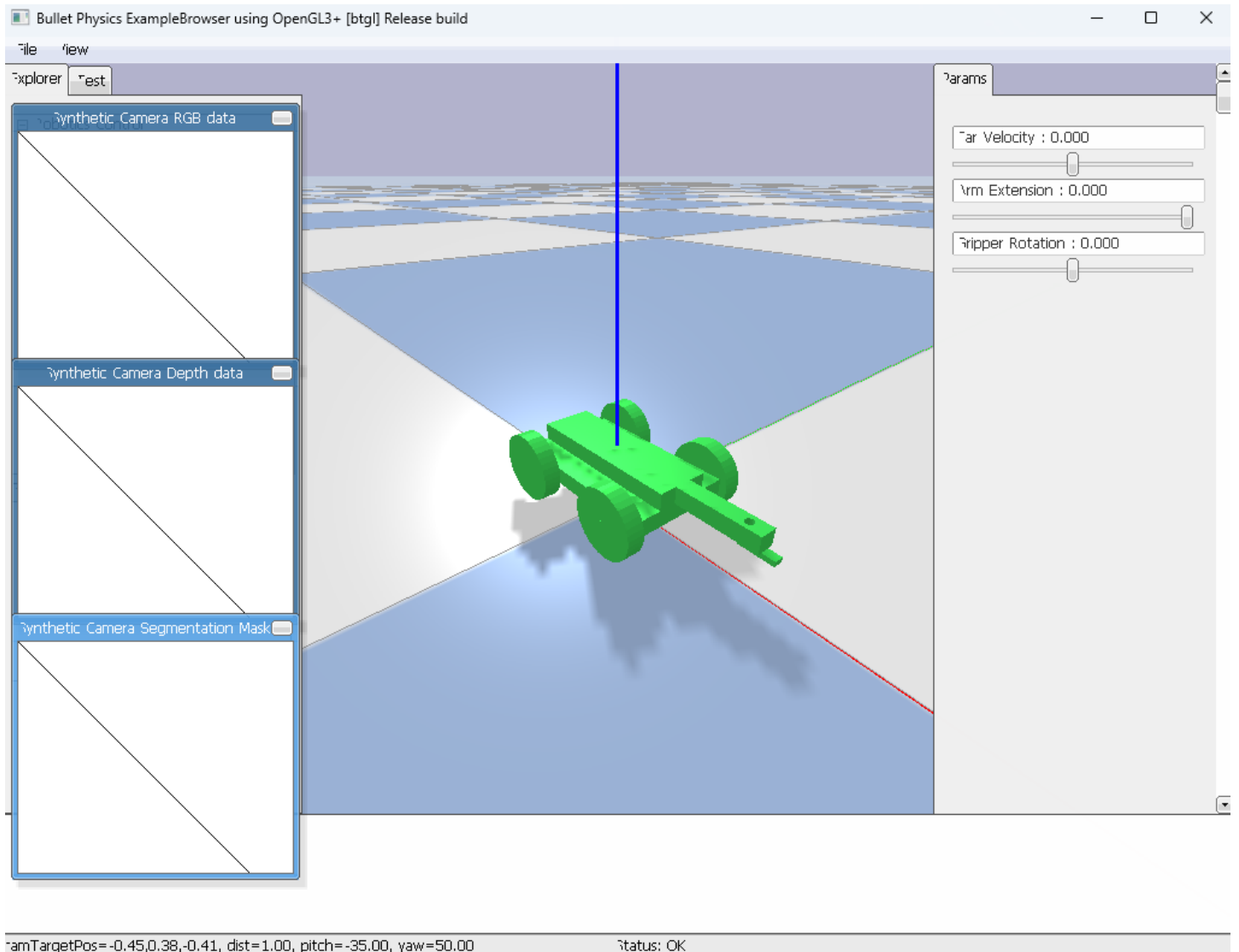
```

```

32 # 关节索引映射
33 wheel_indices = [0, 1, 2, 3] # 四个轮子
34 arm_index = 4 # 伸缩臂
35 gripper_index = 5 # 夹爪
36
37 # 4. 实时仿真循环
38 while True:
39     # A. 读取滑块当前的数值
40     target_velocity = p.readUserDebugParameter(vel_slider)
41     target_arm_pos = p.readUserDebugParameter(arm_slider)
42     target_gripper_pos = p.readUserDebugParameter(gripper_slider)
43
44     for i in wheel_indices:
45         p.setJointMotorControl2(bodyUniqueId=robot_id,
46                                 jointIndex=i,
47                                 controlMode=p.VELOCITY_CONTROL,
48                                 targetVelocity=target_velocity,
49                                 force=10) # 适当给一点力矩
50
51     # C. 控制伸缩杆 (使用位置控制 POSITION_CONTROL)
52     p.setJointMotorControl2(bodyUniqueId=robot_id,
53                             jointIndex=arm_index,
54                             controlMode=p.POSITION_CONTROL,
55                             targetPosition=target_arm_pos,
56                             force=30) # 伸缩需要较大推力
57
58     # D. 控制爪子 (使用位置控制 POSITION_CONTROL)
59     p.setJointMotorControl2(bodyUniqueId=robot_id,
60                             jointIndex=gripper_index,
61                             controlMode=p.POSITION_CONTROL,
62                             targetPosition=target_gripper_pos,
63                             force=20) #
64
65     p.stepSimulation()
66     time.sleep(1./240.) # 维持仿真步长

```

运行程序效果如下：



不知道为什么这个gripper位置反了（= =，摆烂了不管了）

右边有三个滑块分别是速度控制滑块，arm伸缩滑块和gripper转动滑块，读者可以自由拖动试试。

至此，我们顺利地将小车的SW建模导出成urdf，并在pybullet里导入并实现简单的控制。