Infinigen-Sim: Procedural Generation of Articulated Simulation Assets

Abhishek Joshi[1], Beining Han[1], Jack Nugent[1], Yiming Zuo[1], Jonathan Liu[1], Hongyu Wen[1], Stamatis Alexandropoulos[1], Tao Sun[1,2], Alexander Raistrick[1], Gaowen Liu[3], Yi Shao[2], Jia Deng[1]
[1]Department of Computer Science, Princeton University
[2]McGill University
[3]Cisco

Abstract

We introduce Infinigen-Sim, a toolkit for generating realistic, procedurally generated articulated assets for robotics simulation. We include procedural generators for 12 common articulated object categories along with high-level utilities for use creating custom articulated assets in Blender. We also provide an export pipeline to integrate the resulting assets along with their physical properties into common robotics simulators. Experiments show that assets sampled from these generators are useful for movable object segmentation, training generalizable reinforcement learning policies, and sim-to-real transfer of imitation learning policies.

1. Introduction

Interacting with articulated objects is an essential step in many important applications for robotics. From using hinged fridges and dishwashers, pushing buttons, opening drawers, to traversing doors with various handles, robots require the ability to manipulate articulated objects to accomplish useful tasks in the real world.

Learning in physical simulation [1, 5, 10, 20, 26, 34] has proven a useful strategy for many robotics tasks [2, 8, 15, 16, 21, 21, 28, 32]. To best use this strategy for articulated object manipulation, we require large-scale datasets of articulated assets with as diverse geometry, joints, and visual appearance as possible. Articulated assets are challenging to acquire, as objects curated from the internet [4, 6] or 3D-scanning [30] are typically static and lack joint annotations. Widely used articulated object datasets instead use labor-intensive human annotation [11, 14, 19, 24, 31] for joint position and axes, which limits the number and quality of unique geometries and articulation annotations.

To this end, we propose Infinigen-Sim, a tool which enables the creation of high quality, diverse articulated assets via procedural generation. Our tool has many advantages:

1. *Unlimited kinematic variation:* Procedural generators can sample assets with dense coverage of important physical dimensions which impact a robot's trajectory, e.g. the handle-to-hinge distance of a door. Continuous variation also applies to geometry detail (e.g. edge bevel radii) and tolerances (e.g. toaster slot width), which all serve to create dense coverage and diverse training data.
2. *High-quality assets:* Joint locations, axes, and annotations can be verified for correctness unlike previous efforts which rely on human annotation or generative methods. Our assets also utilize physics-based rendering materials present in Infinigen [22, 23] along with material physical properties such as friction and density.
3. *Combinatorial coverage:* Procedural generators can randomly swap sub-parts to use different procedural generators (e.g. changes in handle style: knobs, levers, crash-bars). We can also easily express variable-quantities of articulated parts (e.g. 1-5 fridge drawers), including very small parts (e.g. tiny buttons on a toaster), which are challenging for annotators or image-based capture.

Experiments show that objects from procedural generators created with Infinigen-Sim are useful for tasks in both vision and robot learning. We use Infinigen-Sim to create articulated procedural generators for 12 object categories. We hope Infinigen-Sim will provide an ever-growing library of open-source articulated object generators, as to enable more robust robot learning across many tasks using articulated objects.

2. Infinigen-Sim

Infinigen-Sim consists of a variety of procedural generators for common articulated object categories along with utilities designed to aid in creating custom articulated procedural generators. We also develop export code which produces a file for use in robotics simulators. The overall workflow is visualized in Figure 2.

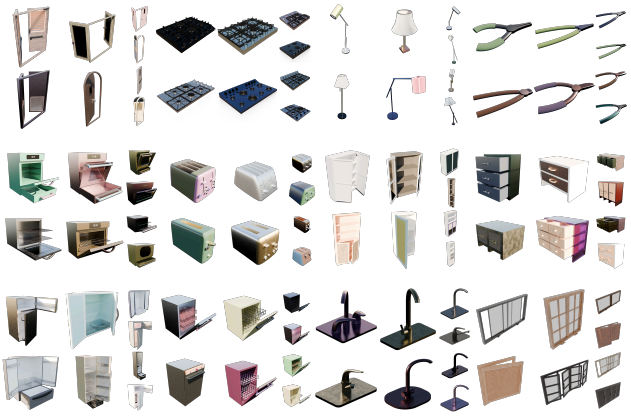

Figure 1. Procedurally generated assets for 12 common articulated object categories.

2.1. Articulated Procedural Generator Tools

Infinigen-Sim leverages Blender's procedural modeling tools [3], namely *Geometry Nodes* and *Shader Nodes*, which are an artist-friendly, GUI-based, domain specific language for procedural generation. These tools allow users to create meshes and materials by composing nodes representing primitives, geometric transformations, and scalar vector arithmetic in a directed acyclic graph. These node-based tools are general purpose and can represent essentially all desirable object geometries.

To design articulated assets within this system, we create two new nodes that implement revolute and prismatic joints. These nodes accept two incoming geometries (defined via prior nodes) and introduce a parent-child relationship annotated with the relevant joint type. Each joint node also allows users to set the pivot position, axis, and joint range, either as fixed values or through parameters passed from other nodes. For the provided procedural generators, we leverage Blender's built-in math nodes to compute these values directly from the asset's geometry within the node graph.

These joint nodes have four purposes. First, they directly speed up the procedural generator creation process since they implement a commonly used operation which no longer needs to be re-implemented for each object that uses a hinge or sliding joint. Second, they provide immediate visual feedback as to the effect of a joint: a hinge joint node with a certain origin or angle extent input will immediately transform the child geometry in the GUI according to those inputs, which prevents erroneous values and can be used to debug any arithmetic expressions responsible for calculating the joint parameters. Third, this standardized implementation of joints allows us to add logic to save all joint metadata in a standard format for later use. Fourth, these nodes can store semantic metadata, ensuring consistent, fine-grained joint and geometry naming across all procedurally generated assets within a category.

Our joint nodes support the creation of many articulated structure types such as jointing multiple meshes together in a chain, jointing multiple meshes to one parent, and adding multiple degrees of freedom between two geometries. We also create an additional custom node group that supports duplicating jointed bodies at defined points. This can be effective for procedurally duplicating articulations such as burners on a cooktop or shelves in a dishwasher without having to manually define each joint. Finally, we provide a utility script that checks if any two rigid parts of the asset penetrate each other within a defined range of motion. This helps users to adapt joint configurations and parts' geometries to avoid self-collision and penetration problems.

2.2. Exporting Assets to Simulation

Generating instances of procedurally articulated assets from Blender to a simulation-ready format involves three steps, all of which are automated. First, we use Infinigen's [22] transpiler to convert the procedural asset's node graph to Python code. Users can edit this code to customize asset parameter distributions or further tune node graph structures. Thus, our tool gives users fine-grained control over generated assets. It also allows users to create dynamic graphs that may change based on the procedural parameters.

Second, we sample random values for each procedural parameter and spawn an instance of the asset, along with its node graph, in Blender. We then use the Blender Python API to parse and update the asset's node graph. During this step, we inject into the graph Blender-provided nodes which assign values for attributes to different parts of the geometry. These attributes enable us to construct a kinematic tree for the asset instance (inspired by [9, 13]) that abstracts away the full procedural graph while preserving the essential details required for articulation.

The final step is to export the asset into a simulation-compatible format. We pass the asset instance along with its corresponding kinematic tree to a native exporter for either one of the URDF, USD, and MJCF file formats. To construct the asset, the exporter recursively traverses the kinematic tree in a depth-first fashion. To get individual rigid bodies, the exporter queries the asset instance with attributes and their values based on the current path in the tree. We then extract the part of the asset that matches this query and save it as an individual mesh. Users can also generate convex-decomposed collision meshes with third-party tools like CoACD [29]. During this process, we use the kinematic tree to build the final simulation-ready asset including all joints, rigid geometries, and semantic metadata.

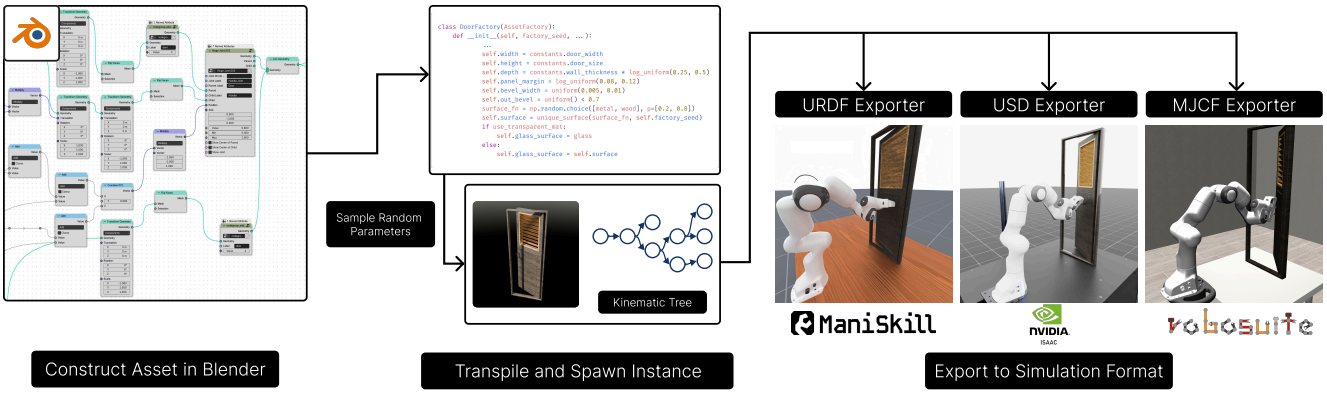

Figure 2. We first create procedural assets using Blender's Geometry Nodes feature. These assets are then transpiled and we have the ability to define custom distributions for the procedural parameters (including materials) along with its dynamics properties. We sample parameters to generate the asset and pass both the asset its kinematic tree to the exporter, which recursively builds a simulation-ready articulated object.

3. Articulated Asset Procedural Generators

Using Infinigen-Sim, we create procedural generators for 12 categories of common articulated assets as shown in Fig. 1. Each procedural generator includes distributions for joint dynamics parameters (i.e. stiffness and damping) and produce kinematically diverse, photorealistic articulated assets. We provide a list of articulated parts for each asset category in 5.1.

4. Experiments

We demonstrate that assets created with Infinigen-Sim help downstream vision and robot learning tasks, including movable part segmentation, RL generalization, and sim-to-real transfer.

4.1. Movable Part Segmentation

Movable part segmentation [27, 31] is an important vision task for embodied agents. Given an RGB image of an object, the model must segment each articulated part. We focus on five categories of assets including doors, toasters, refrigerators[1], dishwashers, and lamps. First, we generate 250 diverse assets per category using Infingen-Sim. Then, we follow a similar setup as [31] to generate image datasets for both PartNet-Mobility and Infinigen-Sim assets. We split the PartNet dataset such that images rendered from 75% of the assets are used for training and the remaining 25% are used for evaluation. All models are evaluated on images of *unseen* PartNet assets only. We finetune a pretrained Mask R-CNN model with a ResNet-50-FPN backbone, as in [12], for three datasets: 1) `P15k`, approximately 15k images of PartNet assets only, 2) `P30k`, approximately 30k images of PartNet assets only, and 3) `P15k+I`, a combination of approximately 15k images of PartNet assets and 15k images of Infinigen-Sim assets.

As shown in Table 1, we observe that doubling the size of the original dataset (`P30k`) yields only marginal improvements, and in some cases decreases performance. This suggests the model may not be learning new features that improve generalization to unseen assets despite scaling the training images from the original PartNet dataset alone. In contrast, we observe an interesting result when adding the same number of images of Infinigen-Sim assets. Specifically, we see a significant improvement across smaller articulated parts such as door handles, toaster levers, dishwasher buttons, and lamp switches. This suggests a core benefit of Infinigen-Sim. Namely, our tool allows users to procedurally scale articulated parts that require a greater level of detail. Achieving this level of granularity is labor-intensive for manually annotated articulation datasets. We additionally note that despite Infinigen-Sim assets being out of distribution compared to the evaluation set, we still see improvements for detailed parts.

4.2. Reinforcement Learning Generalization

We demonstrate that Infinigen-Sim assets help reinforcement learning policies generalize to novel object instances. For each task, we train three policies, namely one trained with only Infinigen-Sim assets, one trained with only Partnet-Mobility assets, and one trained on a combination of both. We evaluate on the following tasks.

1. **Push Door with Handle**. The robot needs to push the door open by first rotating the door handle clockwise by approximately 17 degrees (0.3 radians). Only single, push doors with a lever handle on the left are considered for this task.
2. **Push Down Toaster Lever**. The robot is tasked with

[1]The refrigerators used in the movable part segmentation experiments have since been updated (Fig. 1) with more diverse handle types, more realistic materials, and fewer vertices.

	Door			Toaster				Refrigerator	
Dataset	Frame	Body	Handle	Body	Lever	Knob	Button	Body	Door
P15k	57.41	71.87	41.08	96.96	59.76	55.31	4.20	81.35	66.37
P30k	59.21	**74.65**	40.31	**97.15**	59.05	**57.50**	3.70	82.76	**69.00**
P15k+I	**59.64**	73.17	**44.97**	96.87	**64.31**	54.69	**5.82**	**84.37**	67.82

	Dishwasher					Lamp				mAP
Dataset	Body	Door	Shelf	Button	Knob	Base	Rod	Head	Switch	Overall
P15K	84.23	73.32	0.13	24.59	0.00	54.75	**7.27**	**78.09**	11.45	48.23
P30k	85.14	75.14	**0.21**	21.31	0.00	51.72	7.22	77.86	10.45	48.46
P15k+I	**86.59**	**75.31**	0.17	**31.01**	**0.02**	**55.40**	7.03	77.63	**17.44**	**50.13**

Table 1. Movable part segmentation performance per articulated part type. We train a model for 50 epochs and report the evaluation mAP scores averaged across 3 seeds. Using Infinigen-Sim assets results in stronger model generalization for relatively smaller articulated parts, e.g. door handles, toaster levers, dishwasher buttons, and lamp switches.

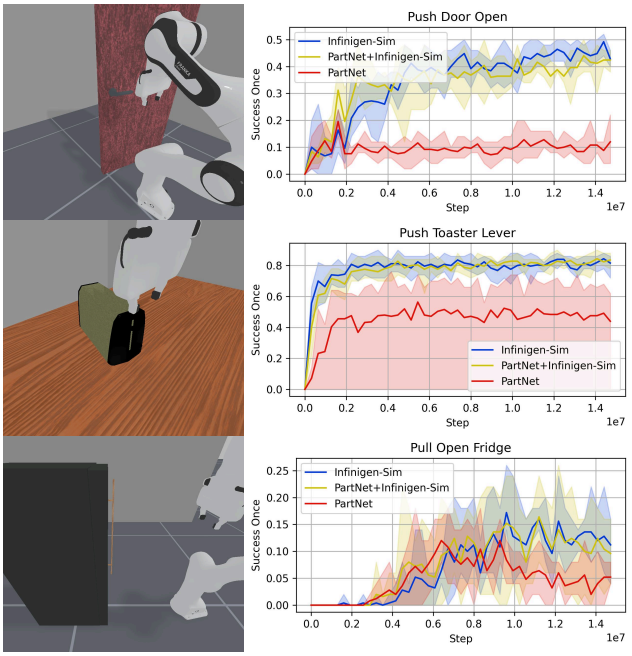

Figure 3. Left: Environment setups for each task. Right: Success once results on evaluation dataset averaged over 5 seeds.

pushing down the lever of the toaster. Only toasters with a single lever are considered for this task.

3. **Open Fridge Door**. The robot is tasked with grasping the handle and opening the fridge door by at least 9 degrees ($\pi/20$ radians). Only refrigerators with a single door with a handle on the left are considered for this task.

Towards the end of training (last approximately 1M timesteps), we see a relative improvement of 4.329, 1.761, 2.5 times for the success rate for the door, toaster, and fridge tasks, respectively, when using Infinigen-Sim assets over PartNet-Mobility. We attribute this performance increase to the scale and quality of Infinigen-Sim assets. PartNet-Mobility assets are limited in quantity, and may have misaligned joints and poor geometries that make it harder to train policies. In contrast, our tool provides dynamically accurate assets with both great visual and geometric diversity. However, it is worth noting that during the earlier stages of training, for the door and fridge opening tasks, the PartNet-Mobility policy performed on par or better than policies trained using Infinigen-Sim. This suggests that realizing the benefits of more diverse assets may require longer training times.

4.3. Sim-to-Real Transfer

We show that Infinigen-Sim assets are effective in sim-to-real transfer. We create real-world versions of the door opening, toaster pushing, and fridge opening tasks using a Franka Panda robot (see Fig. 4). For each task, we collect expert simulation trajectories using cuRobo [25] in an IsaacGym [17] environment. We randomize environment lighting, joint initialization, object position, and camera views. As we have the ground truth joint axis and origin, we rollout scripted trajectories and filter successful ones as the demonstration dataset. The robot takes an RGB image from its on-the-shoulder camera and the end effector pose as inputs. We then train an ACT [33] policy on this dataset and transfer zero-shot it to a physical robot.

	Door	Toaster	Fridge
PartNet-Mobility [31]	6/30	0/30	0/30
Infinigen-Sim (Ours)	**22/30**	**11/30**	**10/30**

Table 2. Zero-shot sim-to-real results.

We observe that policies trained using Infinigen-Sim perform better across all three tasks in the real world (see Ta-

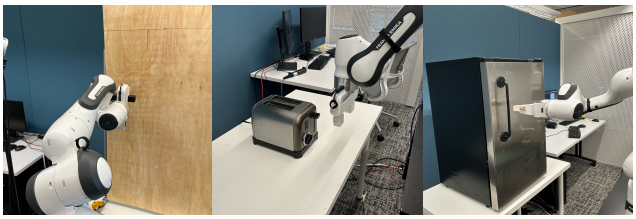

Figure 4. Real world environment setup. From left to right: push open door, push toaster lever, and pull open fridge door.

ble 2). Qualitatively, we notice that policies using our assets tend to be smoother and fail less aggressively. For instance, in the fridge task, our policy's failures typically involved either moving toward the handle and nearly grasping it or opening the door less than the 45 degree threshold. In contrast, policies trained on PartNet-Mobility often missed the handle altogether. Similar to the RL experiments, we attribute this success to the high diversity and quantity of assets generated with our tool. This diversity likely helps bridge the sim-to-real gap and allows the policy to focus on important aspects of the task (e.g. the handle or lever). These experiments show that our tool has the ability to create articulated assets diverse enough to facilitate sim-to-real transfer.

5. Conclusion

We introduce Infinigen-Sim, a toolkit for creating procedurally generated, simulation-ready, articulated assets. These assets are kinematically diverse, photorealistic, and have accurate joint configurations. Our tool also gives users fine-grained control over their assets. We use this system to create expressive generators for 12 common articulated object categories. Our experiments demonstrate that objects from these generators can improve model generalization on both perception and robot learning tasks in simulation and the real world.

References

[1] Genesis Authors. Genesis: A universal and generative physics engine for robotics and beyond, 2024. 1

[2] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025. 1

[3] Blender Online Community. Blender - a 3d modelling and rendering package. https://www.blender.org, 2018. Version 4.2. 2

[4] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 1

[5] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-Scale Embodied AI Using Procedural Generation. In *NeurIPS*, 2022. Outstanding Paper Award. 1

[6] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-xl: A universe of 10m+ 3d objects. *arXiv preprint arXiv:2307.05663*, 2023. 1

[7] Long Le, Jason Xie, William Liang, Hung-Ju Wang, Yue Yang, Yecheng Jason Ma, Kyle Vedder, Arjun Krishna, Dinesh Jayaraman, and Eric Eaton. Articulate-anything: Automatic modeling of articulated objects via a vision-language foundation model. *arXiv preprint arXiv:2410.13882*, 2024. 2

[8] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, 2020. 1

[9] Jiahui Lei, Congyue Deng, William B Shen, Leonidas J Guibas, and Kostas Daniilidis. Nap: Neural 3d articulated object prior. In *Advances in Neural Information Processing Systems*, pages 31878–31894. Curran Associates, Inc., 2023. 2

[10] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Elliott Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, Andrey Kurenkov, Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In *Proceedings of the 5th Conference on Robot Learning*, pages 455–465. PMLR, 2022. 1

[11] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, Mona Anvari, Minjune Hwang, Manasi Sharma, Arman Aydin, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Silvio Savarese, Hyowon Gweon, Karen Liu, Jiajun Wu, and Li Fei-Fei. BEHAVIOR-1k: A benchmark for embodied AI with 1,000 everyday activities and realistic simulation. In *6th Annual Conference on Robot Learning*, 2022. 1

[12] Yanghao Li, Saining Xie, Xinlei Chen, Piotr Dollar, Kaiming He, and Ross Girshick. Benchmarking detection transfer learning with vision transformers, 2021. 3

[13] Jiayi Liu, Hou In Ivan Tam, Ali Mahdavi-Amiri, and Manolis Savva. Cage: Controllable articulation generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17880–17889, 2024. 2

[14] Liu Liu, Wenqiang Xu, Haoyuan Fu, Sucheng Qian, Qiaojun Yu, Yang Han, and Cewu Lu. Akb-48: A real-world articulated object knowledge base. In *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 14809–14818, 2022. 1, 2

[15] Yecheng Jason Ma, William Liang, Hungju Wang, Sam Wang, Yuke Zhu, Linxi Fan, Osbert Bastani, and Dinesh Jayaraman. Dreureka: Language model guided sim-to-real transfer. In *Robotics: Science and Systems (RSS)*, 2024. 1

[16] Abhiram Maddukuri, Zhenyu Jiang, Lawrence Yunliang Chen, Soroush Nasiriany, Yuqi Xie, Yu Fang, Wenqi Huang, Zu Wang, Zhenjia Xu, Nikita Chernyadev, Scott Reed, Ken Goldberg, Ajay Mandlekar, Linxi Fan, and Yuke Zhu. Sim-and-real co-training: A simple recipe for vision-based robotic manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, Los Angeles, CA, USA, 2025. 1

[17] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021. 4

[18] Zhao Mandi, Yijia Weng, Dominik Bauer, and Shuran Song. Real2code: Reconstruct articulated objects via code generation. *arXiv preprint arXiv:2406.08474*, 2024. 2

[19] Yongsen Mao, Yiming Zhang, Hanxiao Jiang, Angel Chang, and Manolis Savva. Multiscan: Scalable rgbd scanning for 3d environments with articulated objects. *Advances in neural information processing systems*, 35:9058–9071, 2022. 1

[20] Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6): 3740–3747, 2023. 1

[21] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems*, 2024. 1

[22] Alexander Raistrick, Lahav Lipson, Zeyu Ma, Lingjie Mei, Mingzhe Wang, Yiming Zuo, Karhan Kayan, Hongyu Wen, Beining Han, Yihan Wang, Alejandro Newell, Hei Law, Ankit Goyal, Kaiyu Yang, and Jia Deng. Infinite photorealistic worlds using procedural generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12630–12641, 2023. 1, 2

[23] Alexander Raistrick, Lingjie Mei, Karhan Kayan, David Yan, Yiming Zuo, Beining Han, Hongyu Wen, Meenal Parakh, Stamatis Alexandropoulos, Lahav Lipson, Zeyu Ma, and Jia Deng. Infinigen indoors: Photorealistic indoor scenes using procedural generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21783–21794, 2024. 1

[24] Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia, Kent Elliott Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, Karen Liu, Silvio Savarese, Hyowon Gweon, Jiajun Wu, and Li Fei-Fei. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Proceedings of the 5th Conference on Robot Learning*, pages 477–490. PMLR, 2022. 1

[25] Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, Nathan Ratliff, and Dieter Fox. curobo: Parallelized collision-free minimum-jerk robot motion generation, 2023. 4

[26] Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint arXiv:2410.00425*, 2024. 1

[27] Ruiqi Wang, Akshay Gadi Patil, Fenggen Yu, and Hao Zhang. Active coarse-to-fine segmentation of moveable parts from real images. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part XXXIV*, page 111–127, Berlin, Heidelberg, 2024. Springer-Verlag. 3

[28] Adam Wei, Abhinav Agarwal, Boyuan Chen, Rohan Bosworth, Nicholas Pfaff, and Russ Tedrake. Empirical analysis of sim-and-real cotraining of diffusion policies for planar pushing from pixels, 2025. 1

[29] Xinyue Wei, Minghua Liu, Zhan Ling, and Hao Su. Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search. *ACM Transactions on Graphics (TOG)*, 41(4):1–18, 2022. 2

[30] Tong Wu, Jiarui Zhang, Xiao Fu, Yuxin Wang, Liang Pan Jiawei Ren, Wayne Wu, Lei Yang, Jiaqi Wang, Chen Qian, Dahua Lin, and Ziwei Liu. Omniobject3d: Large-vocabulary 3d object dataset for realistic perception, reconstruction and generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1

[31] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 3, 4, 2

[32] Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A. Kahrs, Carlo Sferrazza, Yuval Tassa, and Pieter Abbeel. Mujoco playground: An open-source framework for gpu-accelerated robot learning and sim-to-real transfer., 2025. 1

[33] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023. 4

[34] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, Yifeng Zhu, and Kevin Lin. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020. 1

Infinigen-Sim: Procedural Generation of Articulated Simulation Assets

Supplementary Material

5.1. Articulated Parts

We provide a list of articulated parts in Table 3 for each of our procedural generators. Note that certain parts may have their own procedural parameters including style, count, positioning, dimensions, etc. The articulated part may also have multiple degrees of freedom relative to its parent asset (e.g., a soap dispenser nozzle is connected to the base by both a revolute and sliding joint).

Object Category	Articulated Parts
Door	Door, handle
Cooktop	Burner dial
Lamp	Lamp arms, shade, switch/button/dial/pull chain
Oven	Door, racks, drawer
Toaster	Lever, buttons, knobs
Cabinet	Doors
Drawer	Doors
Refrigerator	Doors, internal drawers, external drawers
Dishwasher	Door, buttons, racks, knobs
Faucet	Spout, handles
Window	Window panes, locks
Plier	Plier hand

Table 3. Provided object categories and their articulations.

5.2. Example Articulations

In this section, we showcase minimal examples of how our custom joint nodes can be used to create different types of articulations within Blender. These articulated assets directly translate to simulation using our native exporters for the URDF, USD, and MJCF file types.

Figure 5. **Simple Revolute Joint**: Demonstrates a rod rotating about a pivot point using our custom hinge joint node.

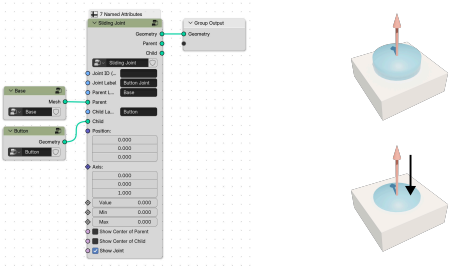

Figure 6. **Simple Prismatic Joint**: Demonstrates an articulated button using our custom sliding joint node.

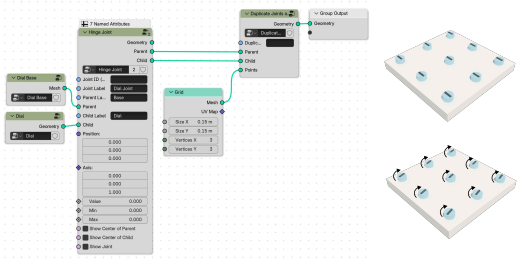

Figure 7. **Duplicating Jointed Bodies**: Demonstrates duplication of multiple articulated knobs. Instead of manually defining a joint for each knob, this node automatically replicates the articulated body at a set of points (a grid in this example).

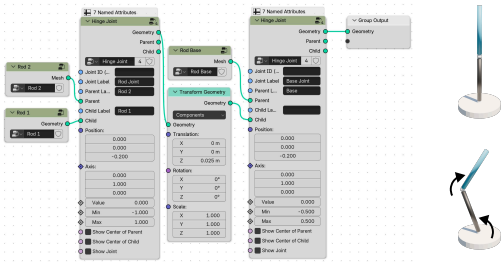

Figure 8. **Connecting Multiple Jointed Bodies**: Demonstrates connecting multiple bodies together. We first joint the two rods together, followed by jointing the combined body with the base.

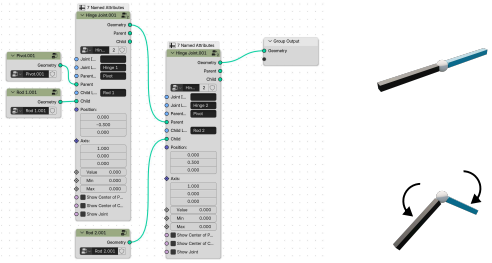

Figure 9. **Jointing to the Same Body**: Demonstrates jointing two rods to the same sphere. First, one rod is jointed to the sphere to form a body. Then, the second rod is jointed to this body by attaching it to the top-most parent object (i.e. the sphere).

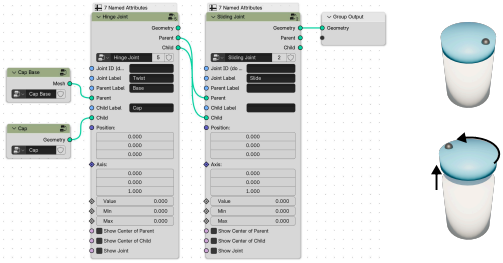

Figure 10. **Multi-Jointed Bodies**: Demonstrates a multi-jointed articulated cap. Combining a hinge and slide joint can achieve a screw joint commonly used for assets such as water bottles.

5.3. Variability in Hinge-to-Handle Distance

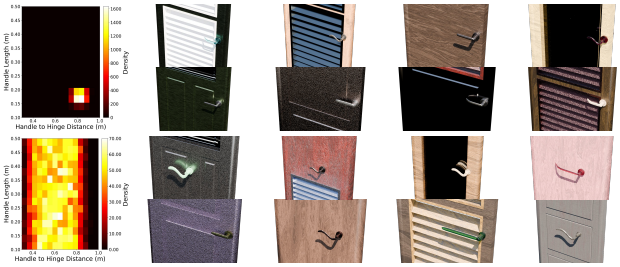

Figure 11. Comparison of asset parameter distributions. The top row shows the default distribution used for handle length and handle-to-hinge distance. The bottom row shows an expanded distribution where these parameters are varied over a wider range.

Assets created using Infinigen-Sim can have many continuously varying input parameters, which can either be manually set or randomized according to a distribution. We show a case study for just doors with lever handles in Fig. 11. To open a door with a handle, two parameters are important for manipulation: the length of the handle and the distance between the handle hinge and the door hinge. The former defines grasp poses and the motion of handle rotation. The latter determines the trajectory required to successfully rotate the door. Infinigen-Sim allows users to define widely distributed combinations of these two parameters with diverse handle geometries. Users can tailor distributions of parameters to their specific needs. Such properties cannot be satisfied with existing articulated assets [14, 31] or modeling tools [7, 18].