# Geometry as an Executable Specification: Procedural Construction of Articulated and Static Objects in Blender

Flyxion
Independent Researcher

November 2026

**Abstract**

A three-dimensional model is usually kept as the result of its construction, a collection of vertices, faces and transforms from which the reasons for their placement have largely disappeared. This essay examines the alternative in which the primary artifact is a program that builds the object, states the conditions under which its parameters describe an object at all, and tests what it has built. The starting point is a repository of twelve Python scripts that generate animated scenic mechanisms in Blender without interactive modeling, among them cams, folding screens, an expanding ring, an iris, intermittent drives and straight-line linkages. All twelve were executed for this study with their final render step removed. Their structure suggests a formal description in which a generator maps parameters and configurations to scenes, and in which a static object is the case with a single configuration. The description is then applied to four new generators for a bookcase, a spiral staircase, a geodesic dome and a hollow vessel. Writing further geometry proves comparatively easy. Establishing that it is correct is harder, and the essay separates assertions that restate a construction from tests that could contradict it. A deliberately introduced defect, a single raised shelf, passed an initial set of volume, extent and interpenetration tests, and was detected only after the intended property was stated. The inquiry then follows the construction outward. Three bookcases composed into a room pass their own tests while standing in the wrong places, which only requirements stated for the room detect. A labelled image export from twenty-four room variants is compared with ray casting, and the comparison exposes an error in the export that neither output revealed alone. An existing family of seeded object generators is examined at the boundary where an engine-independent description becomes a Blender file. Passing any of these tests establishes the tested property and nothing about manufacture, load, safety or usefulness for training a model.

**Keywords:** procedural modeling; Blender; mechanical linkages; parametric design; geometric verification; composition; synthetic data; constructive specification

# 1 Objects Kept as Their Construction

Interactive modeling leaves behind a shape. A shelf in a saved scene has coordinates, and a person looking at it may infer that it was meant to sit midway between two others, but the file need not say so, and nothing objects when the shelf is nudged. A construction script can say more. It names the quantities from which the coordinates follow, it fixes the order in which parts depend on one another, and it can refuse inputs for which the intended object does not exist. The distinction is old in solid modeling, where Requicha's survey separated the physical solid being modeled from the representations that stand for it and asked which representations are valid, unambiguous and complete [9]. Procedural modeling in computer graphics has shown how far rule systems can be carried in producing large families of buildings from compact descriptions [7]. The present concern is narrower and more domestic: what a modest collection of plain Python scripts already does, and what would be needed to extend it to ordinary objects without losing track of what has been shown about them. The work follows an earlier textbook that treats Blender as a workshop for making a situation available for investigation, with geometry built by script, checked by measurement and recorded so that it can be rebuilt [2].

The practical motive is asset production. Interactive modeling yields the object that was worked on, whereas a script can be asked for further members of a family, run unattended and corrected in one place.

A Blender file produced by such a script is one realization. The script, together with its admissible parameters, describes the realizations that could have been produced. This is a useful difference only if the description is executable, so that the claim can be tried, and only if its range is stated, since a program that accepts every number and returns intersecting boards for some of them describes a family that is partly nonsense. Blender is suitable for the experiment because its Python interface exposes mesh data, object hierarchies, keyframes and rendering to a script run without a graphical session [1].

# 2 The Existing Mechanisms

The repository [4] holds two bodies of work. The earlier one, now under `prototype`, is a small benchmark of thirteen construction and editing tasks with a numerical evaluator, calibrated against deliberately defective records. The later one, added in November 2026, is a set of scenic mechanisms: ten directories containing twelve scripts and about 2,600 lines of Python. Several follow entries in Nguyen Duc Thang's catalogue of animated mechanisms [8], adapted as set pieces. Figure 1 shows the preview images stored with the scripts.

Table 1 records an execution of every script under the `bpy` 5.3.0 module, with the final still render removed and all construction, keyframing, assertions and saving retained. Every script completed. The counts describe the resulting scenes and the source text; they measure neither mechanical quality nor visual quality, and the animations were not inspected frame by frame for this study.

## 2.1 Motion Solved and Then Baked

The scripts do not ask a physics engine to discover how their parts move. Each computes the pose of every moving part from the geometry of the mechanism at each sampled instant

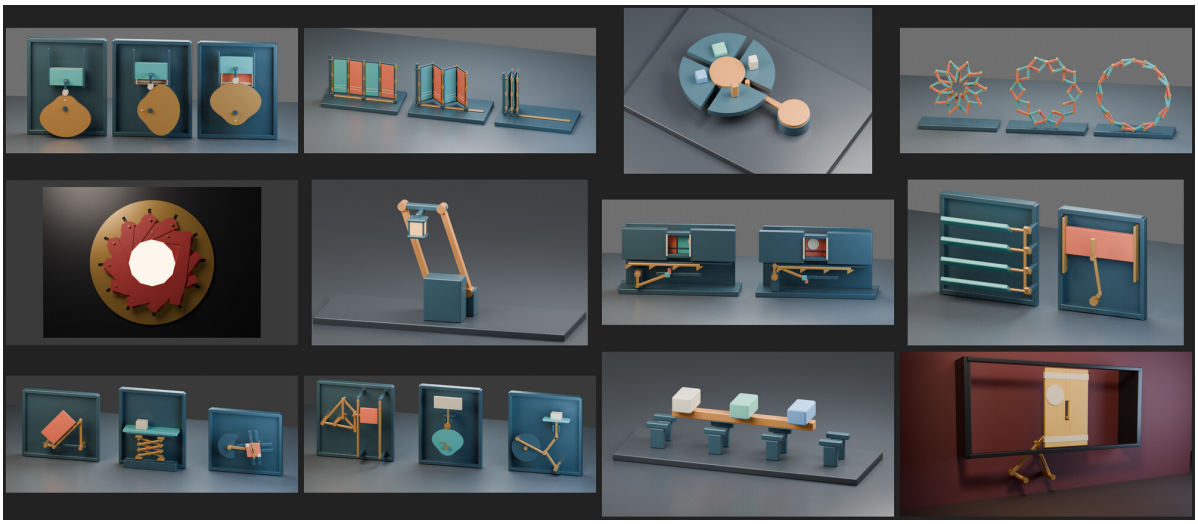

Figure 1: Preview renders stored in the repository for the twelve mechanism scripts.

and writes the result as a keyframe. In the folding stage the hinge points of four equal leaves follow from one opening angle $\alpha$, with successive joints advancing by $\ell \cos \alpha$ along the track and alternating between offsets of zero and $\ell \sin \alpha$ from it, so that every second joint remains on the track where its carriage runs. In the wing shuttle the crank, coupler and rocker are located by intersecting two circles, the coupler is extended to twice its length in the manner of a Hoeken linkage, and the wing takes only the horizontal coordinate of the foot through a vertical slot. The script reports that the foot path is nearly flat over 49 percent of the loop, with a height variation of 0.0049 scene units, while the speed along that stretch varies by 50.2 percent of its mean. The measurement matters because a straight path is easily mistaken for uniform motion.

The intermittent drives show how a few lines of geometry replace manual animation. In the single-mechanism Geneva turntable, with centre distance $d$, pin radius $r = d/\sqrt{2}$ and driver angle $\theta$, the wheel angle during engagement is

$$\psi(\theta) = \operatorname{atan2}\!\left(r \sin \theta,\; d - r \cos \theta\right) + \frac{\pi}{4}, \qquad |\theta| \le \frac{\pi}{4}, \tag{1}$$

and is held constant outside that interval, which gives a quarter turn followed by a dwell. The choice $r = d \sin(\pi/4)$ is the usual condition for a four-slot wheel under which the pin enters a slot along its direction. The cam stage prescribes a follower displacement that rises as $u - \sin(2\pi u)/2\pi$ for normalized time $u \in [0, 1]$, so that velocity and acceleration vanish at both ends of the rise, and then derives the cam outline by offsetting the path of the roller centre inward along its normal by the roller radius.

## 2.2 What the Assertions Establish

The per-frame assertions are of two kinds that are easy to confuse. Some restate the construction. A joint obtained as the intersection of two circles will lie at the two prescribed distances unless the arithmetic is broken, so asserting those distances guards against coding errors and says little about the design. Others could contradict the design. The iris computes each drive pin from the pose of its blade and separately computes the centreline of the slot in the rotating ring, then

| Script | Mesh objects | Animated objects | Frames | Assert lines |
|---|---|---|---|---|
| cam stage | 52 | 15 | 160 | 3 |
| folding stage | 130 | 36 | 160 | 2 |
| Geneva turntable | 16 | 2 | 144 | 0 |
| Hoberman ring | 214 | 150 | 160 | 4 |
| iris shutter | 39 | 25 | 240 | 2 |
| parallel-motion lantern | 19 | 3 | 120 | 1 |
| scenic rack | 65 | 14 | 192 | 3 |
| stage mechanisms, first bank | 45 | 18 | 120 | 2 |
| stage mechanisms, second bank | 77 | 34 | 240 | 9 |
| stage mechanisms, third bank | 64 | 27 | 240 | 12 |
| walking beam | 22 | 4 | 120 | 0 |
| wing shuttle | 21 | 7 | 240 | 4 |

Table 1: Execution record. Assert lines are textual statements in the source, several of which sit inside loops and are evaluated on many frames.

requires the two to agree and the pin to stay within the slot length. The cam banks compare the roller with sampled points of the rotated outline and fail if the roller cuts into the cam or, in the third stage bank, if it loses contact. The lantern records the smallest sampled clearance between the carried pane and a barrier, 1.143 scene units, and requires it to be positive.

**Proposition 1** (Limit of restated constraints)**.** *Let a pose be computed as a solution of equations $c(x) = 0$, and let a test evaluate $c$ at the computed pose. The test can fail only through numerical error or through disagreement between the solver and the evaluator. It carries no information about any property that is not a consequence of $c(x) = 0$.*

*Proof.* An exact solver returns $x$ with $c(x) = 0$, so an exact evaluation of $c(x)$ succeeds for every property of $x$ left free by those equations. Two poses that both satisfy $c$ and differ in such a property receive the same verdict. □

Collision between arbitrary parts is the plainest example of a property left free. The scripts test particular clearances and contacts, and none of them performs a general interference test over all pairs of moving solids. The Geneva turntable and walking beam contain no assertions at all. The walking beam also illustrates a matter of intent that a test would have to be told about: its loads are deposited one pitch forward and keyed there at the final frame, while the first frame places them at their starting stations, so repeating the clip makes them jump back. That is a reasonable depiction of one transfer cycle and an unsatisfactory seamless loop, and the file alone does not say which was wanted.

## 3 Generators, Configurations and Admissibility

**Definition 1** (Generator)**.** Let $\Theta$ be a space of parameters, $\Theta_a \subseteq \Theta$ those parameters satisfying stated preconditions, and for each $\theta \in \Theta_a$ let $Q_\theta$ be a space of configurations. A *generator* is a map

$$G : \{(\theta, q) : \theta \in \Theta_a,\ q \in Q_\theta\} \longrightarrow \mathcal{S}, \tag{2}$$

where an element of $\mathcal{S}$ is a scene description comprising geometry, transforms, materials, component identities and relations among components.

An animation is a trajectory $q : [0, T] \to Q_\theta$, sampled at frames, with realized scenes $S(t) = G(\theta, q(t))$. A static object is the case in which $Q_\theta$ has one element. The restriction removes the solver, the keyframes and the per-frame tests, and it leaves everything else: parameters, preconditions, construction, identity of parts and verification. This is why much of the mechanism code carries over. It is also why static objects are not automatically trivial. Time was one source of failure among several.

**Definition 2** (Check)**.** A *configuration check* is a predicate $\varphi(\theta, q, S) \in \{\text{pass}, \text{fail}\}$ on admissible parameters, a configuration and a scene. A *trajectory check* is a predicate $\psi(\theta, q(\cdot)) \in \{\text{pass}, \text{fail}\}$ on the parameters and a whole trajectory, and may consult any of its realized scenes. A realization satisfies a set of checks $\Phi$ when every member passes. A defect $\delta : \mathcal{S} \to \mathcal{S}$ *escapes* a set $\Phi$ of configuration checks at $(\theta, q, S)$ when $\delta S$ differs from $S$ in an unintended way and $(\theta, q, \delta S)$ satisfies $\Phi$.

A clearance at one instant is a configuration check, as is a clearance required in a prescribed pose. Continuity between successive frames and the agreement of the last pose of a loop with the first are trajectory checks, since no single scene decides them. For a static object $Q_\theta$ has one element and $q$ may be omitted. The definition makes the scope of a passing result explicit. A realization that satisfies $\Phi$ has the properties in $\Phi$. Whether it has the properties its author had in mind depends on whether those were among them.

# 4 Adding Static Objects

The mechanism scripts are self-contained, which makes each easy to read and run and has led to repetition: all twelve define their own functions for materials, boxes and camera aiming, ten define a pin, and six define a bar-posing routine. The new generators place shared construction and inspection routines in one module and leave each object's script to state its parameters, build its parts and test them. The arrangement is an addition beside the existing directories and requires no change to them. Every part is given a `role` property, so that a shelf remains identifiable as a shelf after it has become a mesh.

Adding an object then follows a fixed sequence. The parameters are declared with preconditions that reject impossible combinations. The parts are built from explicit vertex and face lists, which avoids dependence on the selection state of interactive operators. Each mesh is required to be closed, with every edge shared by two faces, and to have positive signed volume. Properties peculiar to the object are then stated and tested. Only afterwards is the scene saved and, on request, rendered. Figure 2 shows the results.

## 4.1 A Bookcase and a Missing Property

The bookcase has width $w$, height $H$, depth $d$, board thickness $t$ and $n$ intermediate shelves, hence $n + 2$ horizontal boards including top and bottom. Its parameters are admissible when $t > 0$, $d > 0$, $n$ is a nonnegative integer, $w > 2t$ and the clear gap

$$g = \frac{H - (n + 2)t}{n + 1} \tag{3}$$

is positive. The construction is short enough to give whole.

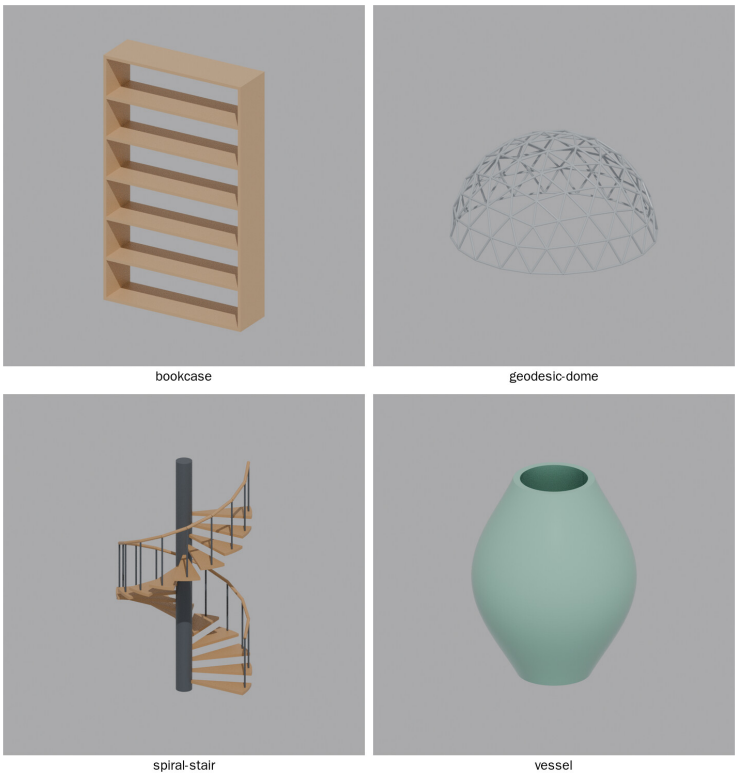


Figure 2: Previews from the four static generators: bookcase, geodesic dome, spiral staircase and vessel.

```
def bookcase(width=1.2, height=2.0, depth=.35, thickness=.025, shelves=5):
    assert thickness > 0 and depth > 0, "boards need positive size"
    assert isinstance(shelves, int) and shelves >= 0, "bad shelf count"
    assert width > 2 * thickness, "sides would meet"
    gap = (height - (shelves + 2) * thickness) / (shelves + 1)
    assert gap > 0, "boards would collide"
    wood = material("Birch", (.55, .34, .16))
    half, t = width / 2, thickness
    for side, x0 in (("Left", -half), ("Right", half - t)):
        box(f"{side} side", (x0, 0, 0), (x0 + t, depth, height), wood, "side")
    for i in range(shelves + 2):
        z = i * (gap + t)
        box(f"Board {i}", (-half + t, 0, z), (half - t, depth, z + t), wood,
            "shelf")
    return gap
```

**Proposition 2** (Board volume)**.** *For admissible parameters the boards have total volume* $V = td\,(2H + (n+2)(w-2t))$ *and pairwise disjoint interiors.*

*Proof.* Each side occupies $t \times d \times H$ and each horizontal board $(w-2t) \times d \times t$, which gives the sum. The sides are separated along the width because $w > 2t$. The horizontal boards lie strictly between them in that direction, and consecutive boards are separated vertically by $g > 0$. □

The script tests part count, closure, volume against this formula, overall extent and pairwise interpenetration, the last being exact because all parts are axis-aligned boxes. All 243 combinations of three values for each of the five parameters passed. Four defects were then introduced into the default object. A shelf shifted sideways was reported as penetrating a side, a removed face left four edges without two faces, and a deleted shelf failed the count. A shelf raised by one hundredth of a scene unit escaped.

**Proposition 3** (Escape of a raised shelf)**.** *Translating one intermediate shelf vertically by $\varepsilon$ with $0 < |\varepsilon| < g$ preserves part count, closure, total volume, overall extent and disjointness of interiors.*

*Proof.* A translation changes neither the number of parts nor the incidence or volume of a mesh. An intermediate shelf is not extremal in any coordinate before or after the move, since the bottom and top boards still bound it, so the extent is unchanged. Its distance to each neighbour remains at least $g - |\varepsilon| > 0$. □

The remedy was to state the property that had been taken for granted, that the clear distances between consecutive boards are equal:

```
# Boards must be evenly spaced. Nothing above detects a raised shelf.
boards = sorted((bounds([o]) for o in parts if o["role"] == "shelf"),
                key=lambda box_: box_[0][2])
clear = [b[0][2] - a[1][2] for a, b in zip(boards, boards[1:])]
assert max(clear) - min(clear) < 1e-5, "uneven shelf spacing"
```

With this test the raised shelf is rejected. The textbook reports a related case from a staged review of fifteen builds, in which the property tests taken together admitted exactly one defective build, a mirrored staircase whose volume, extent and area were correct while it descended where the original rose [2]. The present episode concerns one object and four chosen defects, so it supplies no detection rate. It does show the direction in which a library of generators has to grow, since each new kind of object brings intentions that generic mesh tests do not know.

## 4.2 A Dome Whose Base Depends on Frequency

The dome subdivides each face of an icosahedron into $f^2$ triangles, projects the points to a sphere of radius $R$ and places a strut along each edge whose ends lie on or above the equator, following the spherical icosahedral construction familiar from Fuller's patent [6]. The full sphere has $V = 10f^2 + 2$ nodes and $E = 30f^2$ edges, and the script asserts both before cutting. At $f = 4$ the hemisphere has 91 nodes and 250 struts in six length classes, and every node on the rim lies at height zero. Changing one parameter to $f = 3$ yields 46 nodes, 120 struts in three length classes, and rim nodes at heights 0.3433 and 0.3752 for $R = 2$. The program runs without complaint in both cases and only the first stands on a flat base. A requirement of flatness belongs in $\Theta_a$ or in $\Phi$, and a design for odd frequencies would need a different cut or supports of differing length. Struts are separate closed prisms that overlap at the nodes; no hubs or joints are modeled.

### 4.3 A Staircase and a Vessel

The staircase places eighteen annular sector treads around a column, each rising 0.18 units and sweeping 22.5 degrees, with a baluster on each tread and rail segments between their heads. The script tests closure and equal rise, and reports a going of 0.319 at two thirds of the radius and a clear height between turns of 2.830. The latter is computed from the number of steps per turn, the rise and the tread thickness. It is not a measurement of free space, and neither the meeting of treads with the faceted column nor the overlaps at the rail are tested. No claim of conformity with any building rule follows.

The vessel revolves a single profile, up the outside and back down the inside, into one closed surface with an inner floor. Its mesh volume is compared with the difference of the volumes of two stacks of frusta on the same 128-sided polygon, computed from the profile without reference to the faces. The two agree to within $10^{-4}$, giving a clay volume of 0.17687 and a capacity of 0.69971 cubic scene units. The comparison would expose a wrongly assembled surface. It would not expose an ugly profile, a wall too thin to fire, or a base on which the pot would rock.

## 5 Headless Geometry as Production Infrastructure

A headless script changes who or what can ask for an object. In an interactive session a person arranges a scene and exports it. Run from a command line, a construction accepts parameters, performs its tests, writes its outputs and returns a status, so that another program can call it without knowing how the interface works. The textbook gives the practical reason for building through Blender's data interface in such a setting: operators depend on the current selection and active object, whereas direct data calls change what they are told to change [2]. The mechanism scripts use both, and the static generators written here build meshes from data throughout.

A callable construction needs a contract about failure. The earlier headless suite in the Centerfuge repository supplies one [3]. Each scene writes a record of its generator, commit, parameters, outputs and verification results into a manifest, a separate program checks that manifest without installing Blender, rendered images are rejected when nearly uniform, and scenes that claim to show an interior cast rays to confirm that tagged parts are visible through the opening. A second suite of experiments keeps failed checks as diagnostic artifacts and leaves a separate verifier to give the batch a failing status. The two arrangements differ in where failure is signalled and agree that producing a picture is insufficient evidence of success. The mechanism and static directories discussed here do not yet write such a record, which is the most immediate addition that would let them serve as components of a larger process.

## 6 Objects as Composable Systems

An object that is to be used inside another needs more than its own correctness. The bookcase unit was given a placement operation that builds it, runs its own tests, moves its parts to a position against the back wall of a room and labels them as one unit. The room then states requirements that no unit can state alone: each unit stands on the floor, touches the wall, remains inside the room, keeps clear of the space reserved for a door, and does not overlap another unit. Overlap is judged on the bounding boxes of the units, which is conservative for these shapes.

**Proposition 4** (Placement is invisible to unit checks)**.** *Let every check of a unit be invariant under rigid translation of all its parts. Then no translation of the unit changes the outcome of those checks.*

*Proof.* The statement is the definition of invariance applied to the translated unit. □

The proposition is trivial and its consequence is practical, since part count, closure, volume, extent, mutual interpenetration and shelf spacing are all invariant in this sense. Four placement defects were introduced into an accepted arrangement of three units. A unit was lifted from the floor, pulled away from the wall, pushed into its neighbour, or moved into the door space. In every case the displaced unit passed its own tests and the room reported the corresponding fault. Composition therefore adds obligations at each level, and a component should expose what its container needs in order to state them: its roles, its extent, the surfaces by which it attaches and the space it requires to be used. Only the first two are explicit in the present code. Attachment is performed by arithmetic on known dimensions, and the clearance needed to take down a book is not represented at all.

## 7 Procedural Audiovisual Production

A scene that can be rebuilt can also be filmed again. The repository already contains short looping animations of the iris and the wing shuttle in video and animated image form, produced from keyframes that the scripts computed [4]. The same arrangement suggests supplementary footage and explanatory animations in which a parameter, a camera path or a highlighted component is varied by a line of code, and in which a correction to the geometry can be carried into every shot by rendering again. Component roles would let an editor isolate the follower of a cam or the shelves of a bookcase without painting masks by hand. These are prospective uses. No editing pipeline, camera library or narrated sequence was built for this study, and the cost of rendering long sequences, together with the judgement needed to make them legible, remains. A loop also needs to be declared as such, as the walking beam showed.

## 8 Synthetic Training Environments

A generator knows what it placed. This suggests a source of images with labels for component identity, depth, dimensions and transforms, obtained from the construction instead of from annotation. A small export was written to try the first part of that idea. It samples width, height, depth and shelf count for the three bookcases from a seeded random source, requires each room to pass its checks, and for each of twenty-four variants produces a front orthographic label image of $320 \times 192$ pixels, a depth array and a record of every part with its role, unit, bounding box and visible pixel count. Labels were obtained in two ways. One renders each part with a flat emissive level equal to its number. The other casts a ray through each pixel centre and records the first part hit, together with its distance.

The first complete run disagreed on 203,136 of 1,474,560 pixels. The rendered background was not black. The script had set the world's plain colour, while the renderer used the colour held in the world's node tree, whose default grey was recorded as level 13, the number belonging to one of the bookcase boards. The label images looked like label images, and the records, which were computed from the ray cast, were internally consistent. Setting the node colour

reduced the disagreement to zero. A second run with the same seed produced identical records and depth arrays and identical decoded pixels, although all twenty-four image files differed in their bytes, a distinction between files and pixels that the textbook also reports for repeated renders [2].

Agreement between the two methods shows that they assign the same part to the same pixel in this restricted setting. Both read the same geometry, so agreement cannot show that the geometry or the roles are what was intended, and an orthographic view of flat grey identifiers is far from a photograph. No model was trained. Whether data of this kind improves any perception system depends on variation in materials, lighting, viewpoint and clutter, and on evaluation against real images, none of which is addressed. What the exercise does establish is modest and useful: a second route to the same label can detect an export error that a single route reports with complete confidence.

## 9 Generative Asset Repositories

A repository of generators preserves the capacity to produce objects, where a folder of meshes preserves particular products. The Micromegas package in a further repository is an existing step in this direction [5]. Six seeded generator families describe scenes as trees of named primitives with transforms, materials and metadata, write those descriptions as structured text without needing Blender, and offer a separate backend that creates a Blender file. For this study each family was built twice at seed 1847 and complexity 4, with identical descriptions, and then passed through the backend. Its supplied sweep over twenty seeds reports twenty distinct profiles of primitive counts for five families and nineteen for the sixth, which measures variation in counts and not in shape.

| Family | Primitives | Objects in file | Below a moved ancestor | With metadata |
|---|---|---|---|---|
| great microscope | 99 | 99 | 65 | 52 |
| empty book | 72 | 72 | 0 | 57 |
| Sirian instrument | 74 | 74 | 42 | 1 |
| cosmic necklace | 201 | 201 | 193 | 134 |
| alien measure | 88 | 88 | 38 | 81 |
| scale reliquary | 39 | 39 | 0 | 31 |

Table 2: Micromegas descriptions and their Blender files. In the files no object had a parent, a material or a custom property.

Every primitive arrived in the file (Table 2). The hierarchy, material assignments and metadata did not. The backend also places each object using only its own transform, so that where the transforms of a tree are meant to compose, as the package's nesting operation suggests, objects below a translated, rotated or scaled ancestor are not placed as the description implies. The description remains intact and the loss lies in one adapter, which could parent the objects, assign materials and copy metadata to custom properties. A test comparing the file with the description, object by object, would then belong to the family. The textbook's study of export reaches a compatible conclusion for another boundary, where properties cross only when requested and some mappings are lost [2].

The same sequence used for the static generators admits many further families. Furniture extends the bookcase with joints and openings. Architectural frames, masonry arches and roof

trusses add relations of support. Pipe manifolds, pallets and heat exchangers add connectivity, further profiles give bottles and lampshades, and periodic lattices and parametric shells test the geometric routines. These are proposals. A library could ship each generator with its preconditions, its checks, its interfaces and a set of defects that those checks must catch, and could report the defects that escape. The older benchmark in the repository, whose evaluator was exercised against deliberate defects, and the staged review described in the textbook supply working precedents. Such a record would tell a user what a passing result covers. It would grow more useful as escapes were found and turned into checks, without ever certifying that none remained.

## 10 Limits

The executions reported here used one version of the Blender Python module and omitted the mechanism renders. Tolerances are numerical tolerances on idealized geometry. No script models friction, force, deflection, wear, fastening or material failure, and a mechanism whose keyframes satisfy its linkage equations may still bind, sag or collide when built. The mechanisms were designed as visual studies for set pieces and should not be read as manufacturing or rigging specifications. The static generators, the room and the label export are small, were written for this essay, and have been exercised only as described. The Centerfuge suites are described from their documentation and were not rerun, and results attributed to the textbook are its author's and were not reproduced here. The Micromegas files were inspected for counts, parenting, materials and properties, not for appearance.

## 11 Conclusion

The scenic mechanisms already treat an object as something computed: dimensions determine poses, poses determine keyframes, and a saved scene records one outcome. Static objects fit the same form with the configuration space reduced to a point, and adding them requires little new machinery. What the form offers beyond convenience is a place to write down intentions where they can be tried. A script that says how a bookcase is built is more informative than its mesh, and a script that also says the shelves are evenly spaced is more informative again, because it can now be wrong in a way that someone will notice.

Four failures met along the way have a common shape. A raised shelf satisfied every property first stated, a unit in the doorway satisfied every property of a unit, a grey background produced a well formed label image, and an exported file kept every primitive while losing materials, metadata and parenting. Each became visible when a further statement or a second route exposed a difference. This essay, its source, its programs and its figures are likewise related realizations of one project, none of which contains all of it. A reader can rebuild the figures and rerun the checks, and in doing so learns what was tested. What the author wanted remains available only to the extent that someone wrote it down.

# References


[1] Blender Foundation. *Blender Python API Documentation*.

[2] Flyxion. (2026). *Blender as an Experimental Workshop*. Textbook Manuscript, Revision 1, 13 October 2026. Centerfuge Repository.

[3] Flyxion. (2026). *Centerfuge*. Software Repository, GitHub: standardgalactic/Centerfuge.

[4] Flyxion. (2026). *Blender Experiments*. Software Repository, GitHub: standardgalactic/blender-experiments.

[5] Flyxion. (2026). *Micromegas Procedural Lab*. Software Package in GitHub: standardgalactic/memory.

[6] Fuller, R. B. (1954). *Building Construction*. U.S. Patent 2,682,235.

[7] Müller, P., Wonka, P., Haegler, S., Ulmer, A., and Van Gool, L. (2006). Procedural Modeling of Buildings. *ACM Transactions on Graphics*, 25(3), 614–623.

[8] Nguyen, D. T. (2016). *2100 Animated Mechanical Mechanisms*. Illustrated Catalogue with Videos.

[9] Requicha, A. A. G. (1980). Representations for Rigid Solids: Theory, Methods, and Systems. *ACM Computing Surveys*, 12(4), 437–464.