Ao: Homoiconic Solid Modeling

Matt Keeter | matt.keeter@cba.mit.edu

Massachusetts Institute of Technology

The Center for Bits and Atoms

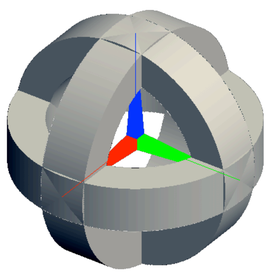
Overview

Ao is an open-source CAD tool for **homoiconic solid modeling**, optimized for **fast CSG** and well-suited for **exploratory and experimental** design.

It includes a geometry kernel (with solids described as **functional representations**) and a set of user-facing tools and libraries.

User-facing	Geometry kernel
Library of primitives: • Basic shapes • Geometric transforms • CSG operations UI with 3D viewport Automatic tracking of bounding boxes	Tracing JIT to convert lambda functions to math trees Fast evaluator Export to various formats: • Mesh (dual contouring) • Heightmap

High-level example



```
; Initial 2D pattern
(define s (difference
          (circle '(0 0) 0.7)
          (circle '(0 0) 0.2)))

; Extruded into 3D
(define e (extrude-z s -0.2 0.2))

; And rotated a few times
(define model (union e
          (rotate-x e (/ pi 2))
          (rotate-y e (/ pi 2))))
```

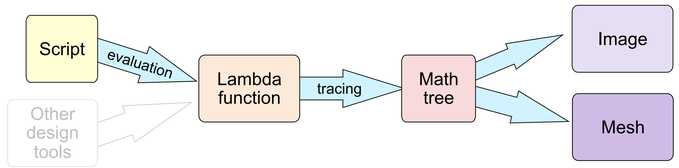
Motivation

- F-rep design offers unique advantages:
  - Easy CSG
  - Continuous deformations and coordinate transforms
  - Easy representations of tiling microstructures
- Large-scale design (10^3 to 10^{6+} clauses) with f-reps is largely unexplored, but CPUs (and GPUs) are now fast enough to make it thinkable!
- The Ao kernel is a stable base for further research:
  - Open source
  - C API
  - Unit tests
  - < 10K well-commented LOC

Homoiconic design

Homoiconic design means that primitive shapes are in the same language as user-created models. The geometry kernel is invisible:

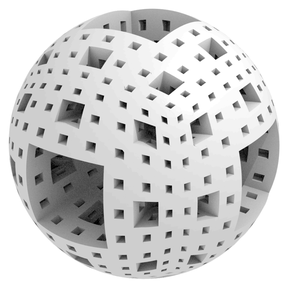
```
(define (sphere r)
    (lambda (x y z) (- (+ (* x x) (* y y) (* z z))
                       (* r r))))
```

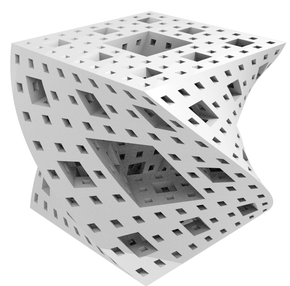


Using Scheme as the host language makes manipulation of shape functions easy. Designers can build shapes and transforms that are hard to express with conventional CAD systems:

```
(define (spherify f) (lambda (x y z)
    (let ((scale (/ (max x y z)
                    (sqrt (+ (* x x) (* y y) (* z z))))))
    (f (/ x scale) (/ y scale) (/ z scale)))))

(define (twist f) (lambda (x y z)
    (let ((a (* (+ z 1) (/ pi 4)))
          (b (atan y x))
          (r (sqrt (+ (* x x) (* y y)))))
   (f (* r (cos (+ a b))) (* r (sin (+ a b))) z))))
```



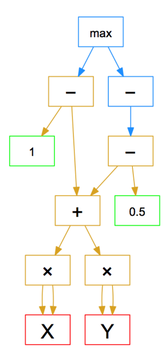


Internal tree representation

Ao's kernel is based on a data structure for fast evaluation of arithmetic trees. DAGs of clauses are sorted topologically and evaluated in order.

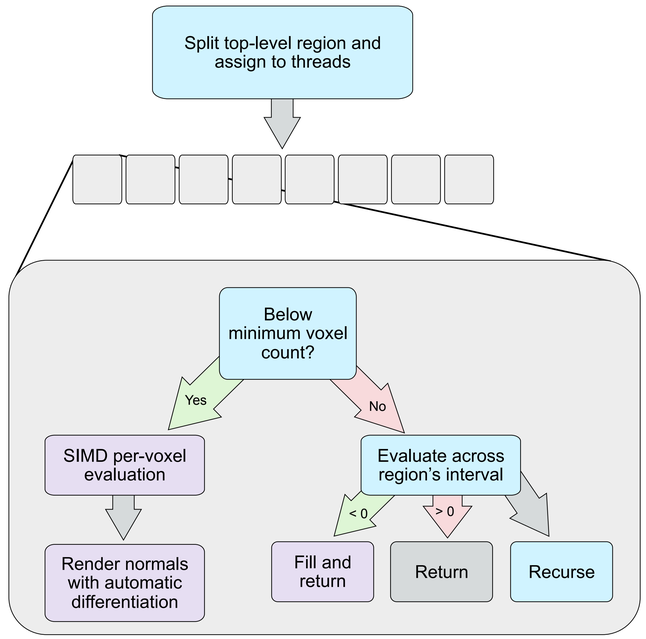
Evaluation can happen over:
- Regions (using interval arithmetic)
- Voxels (with SIMD for multipoint evaluation)
- Gradients (using automatic differentiation)

The tracing JIT evaluates the lambda function with overloaded operations, e.g. **+** builds an addition clause in the tree instead of doing normal addition.



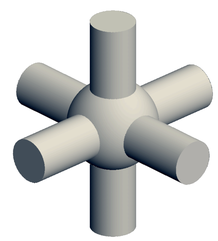
Rendering pipeline

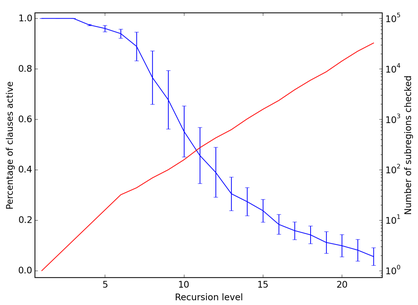
Rendering is similar to Duff '92, updated to take advantage of modern CPUs (in particular, multithreading and SIMD instructions).



Rendering produces a heightmap and shaded image, both pixel-perfect at screen resolution.

A fragment shader on the GPU blends these images together with proper z-culling.





min and **max** clauses are checked after each interval evaluation, and inactive branches are disabled.

For CSG-heavy shapes, this optimization has a huge impact on evaluation speeds.

References

T. Duff, "Interval arithmetic recursive subdivision for implicit functions and constructive solid geometry," in Proceedings of the 19th annual conference on Computer graphics and interactive techniques, ser. SIGGRAPH '92. New York, NY, USA: ACM, 1992, pp. 131–138.

A. Pasko and V. Adzhiev, "Function-based shape modeling: mathematical framework and specialized language," in Automated Deduction in Geometry, Lecture Notes in Artificial Intelligence 2930, 2004, pp. 132–160.

Digital Materialization Group. Hyperfun. [Online]. Available: http://hyperfun.org/