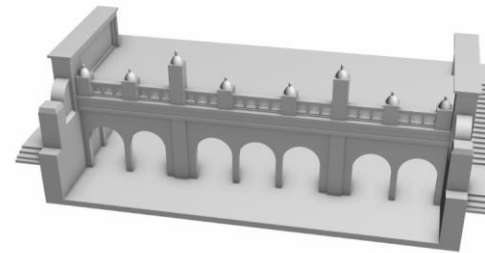


Massively Parallel Rendering of Complex Closed-Form Implicit Surfaces

Matthew J. Keeter

Siggraph 2020



Axel Paris

06/04/2022

Introduction

Analytical implicit surfaces

$$f(x, y, z) = 0$$

```
// Sphere
// p : point
// c : center of sphere
// r : radius
float Sphere(vec3 p, vec3 c, float r)
{
    return length(p-c)-r;
}

// Sphere
// p : point
// n : Normal of plane
// o : Point on plane
float Plane(vec3 p, vec3 n, vec3 o)
{
    return dot((p-o),n);
}

// Operators

// Union
// a,b : field function of left and right sub-trees
float Union(float a, float b)
{
    return min(a,b);
}
```

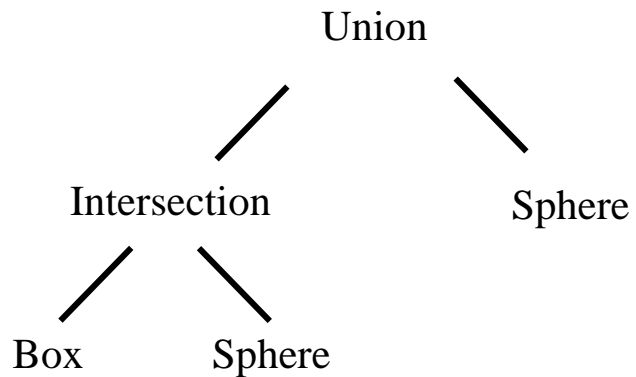
Shadertoy uses them extensively

Introduction

Analytical implicit surfaces

$$f(x, y, z) = 0$$

Construction tree paradigm



```
// Sphere
// p : point
// c : center of sphere
// r : radius
float Sphere(vec3 p, vec3 c, float r)
{
    return length(p-c)-r;
}

// Sphere
// p : point
// n : Normal of plane
// o : Point on plane
float Plane(vec3 p, vec3 n, vec3 o)
{
    return dot((p-o),n);
}

// Operators

// Union
// a,b : field function of left and right sub-trees
float Union(float a, float b)
{
    return min(a,b);
}
```

Shadertoy uses them extensively

```
float evalImplicit(vec3 p)
{
    return min(max(Box(), Sphere()), Sphere());
}
```

Introduction

Interpreter on the GPU

Interval arithmetic

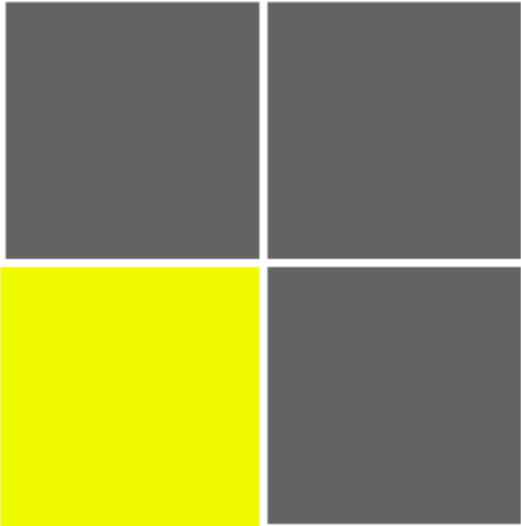
Tape simplification

Full rendering pipeline

Full pipeline

Similar to a **voxelization** algorithm

First pass



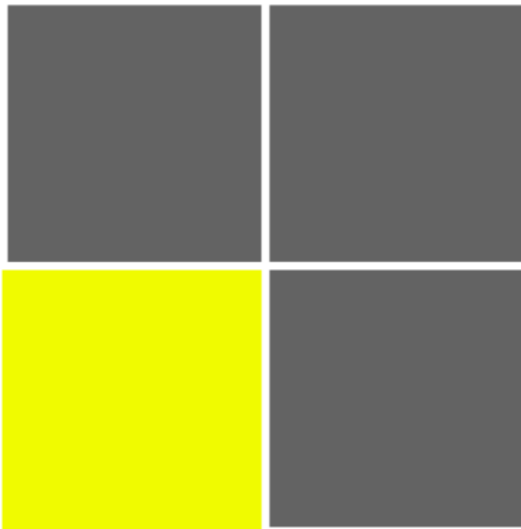
-  64x64 filled tile
-  8x8 filled tile
-  8x8 empty tile
-  8x8 ambiguous tile

Pipeline in 2D

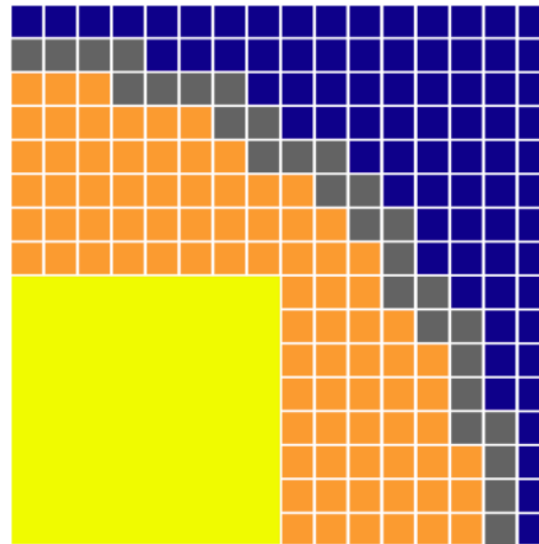
Full pipeline

Similar to a **voxelization** algorithm

First pass



Second pass



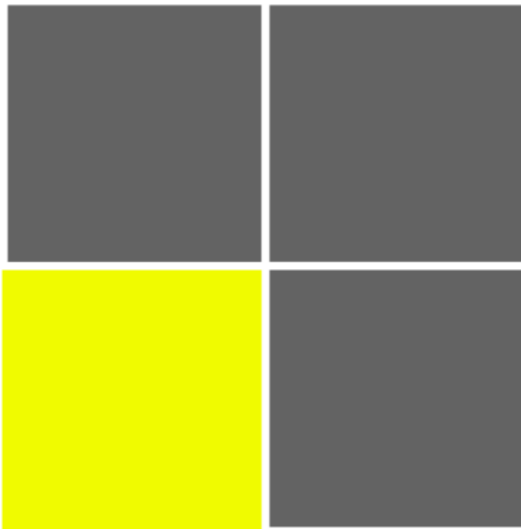
-  64x64 filled tile
-  8x8 filled tile
-  8x8 empty tile
-  8x8 ambiguous tile

Pipeline in 2D

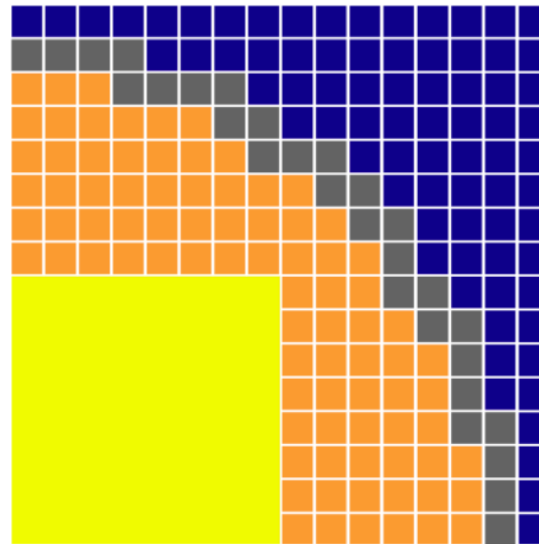
Full pipeline

Similar to a **voxelization** algorithm

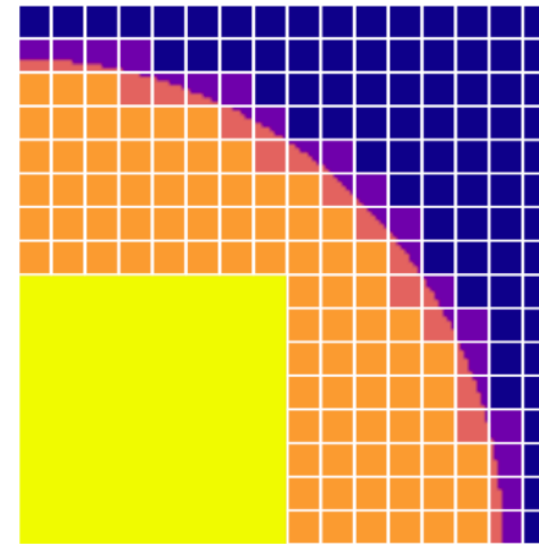
First pass




Second pass



Per-pixel evaluation



-  64x64 filled tile
-  8x8 filled tile
-  8x8 empty tile
-  8x8 ambiguous tile

Pipeline in 2D

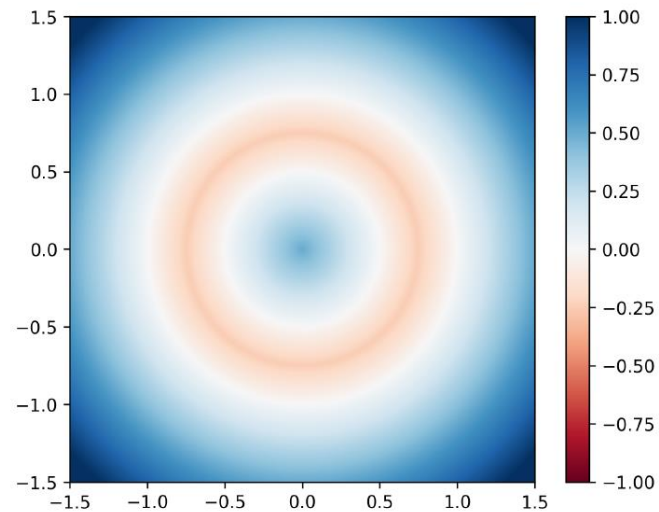
Interpreter on the GPU

Key point: represent the implicit surface as a *tape*

Interpreter on the GPU

Key point: represent the implicit surface as a *tape*

$$\max\left(0.5 - \sqrt{x^2 + y^2}, \sqrt{x^2 + y^2} - 1\right)$$

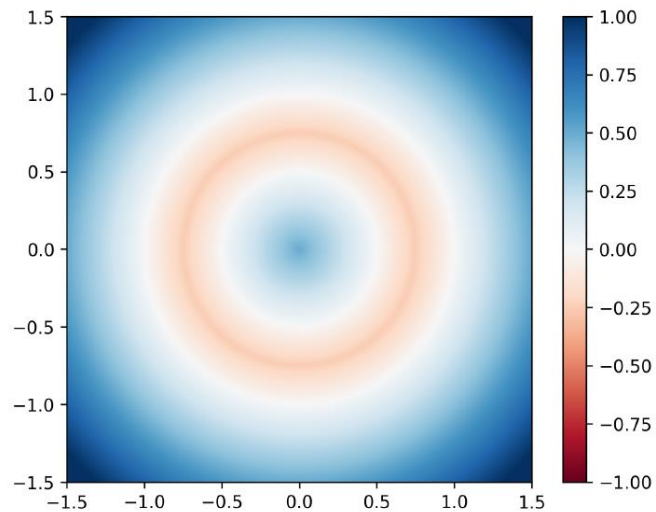


Implicit surface equation

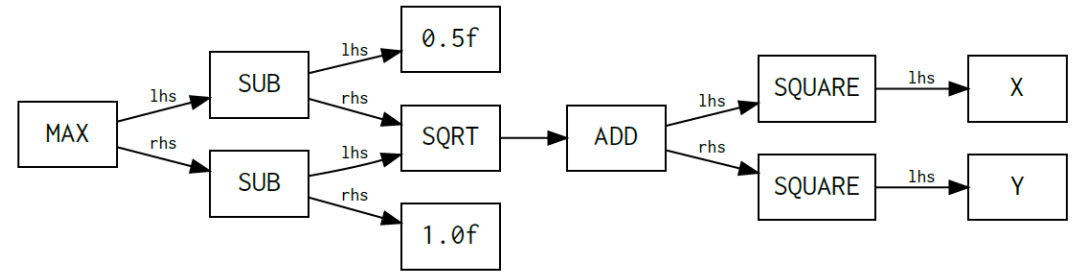
Interpreter on the GPU

Key point: represent the implicit surface as a *tape*

$$\max\left(0.5 - \sqrt{x^2 + y^2}, \sqrt{x^2 + y^2} - 1\right)$$



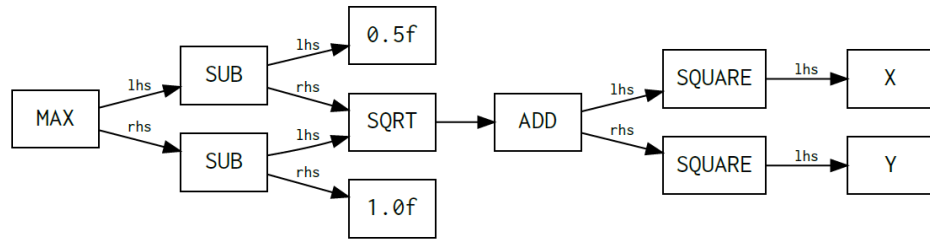
Implicit surface equation



DAG

Interpreter on the GPU

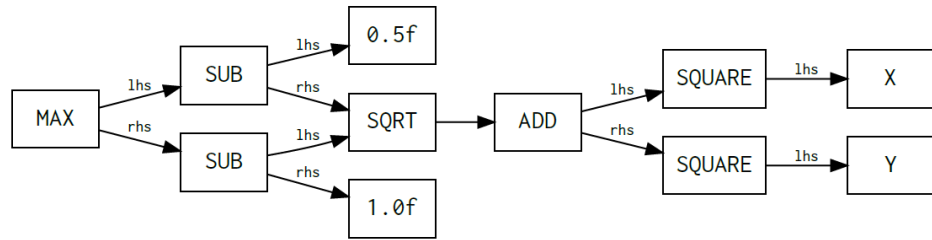
Key point: represent the implicit surface as a *tape*



DAG

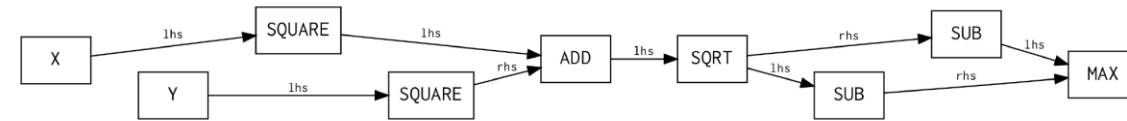
Interpreter on the GPU

Key point: represent the implicit surface as a *tape*



DAG

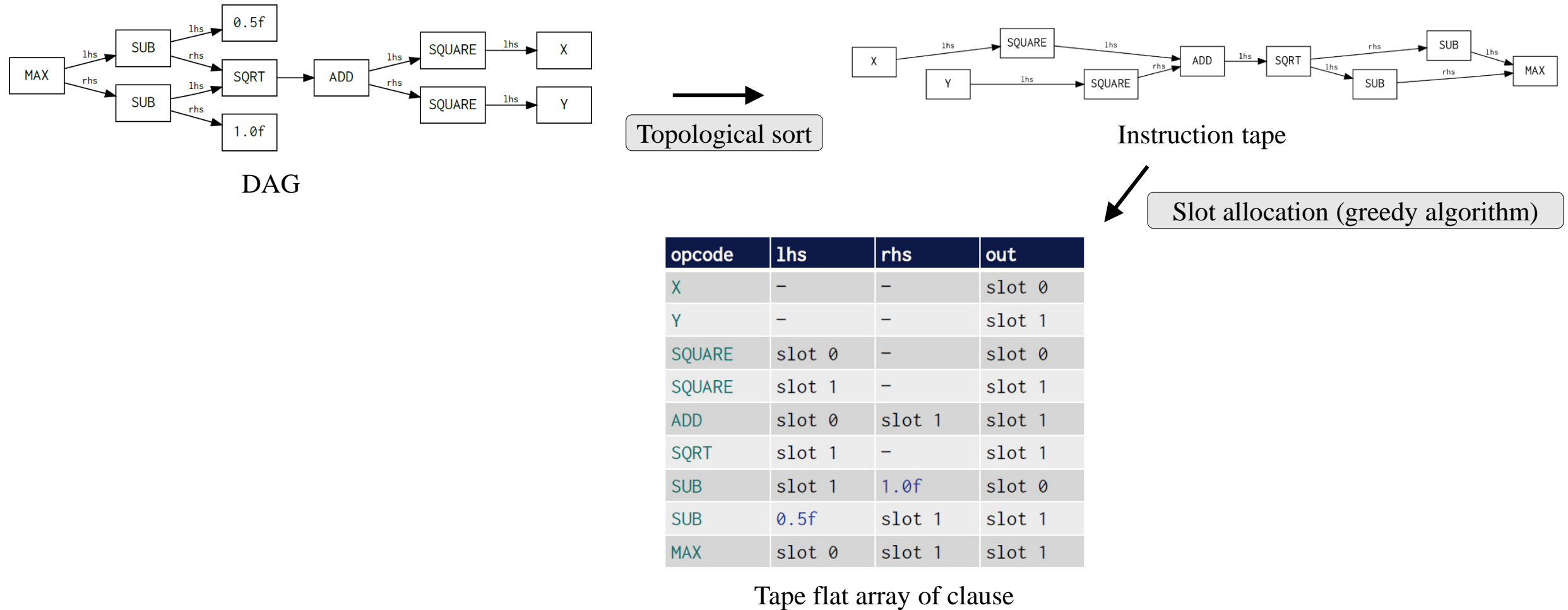
→
Topological sort



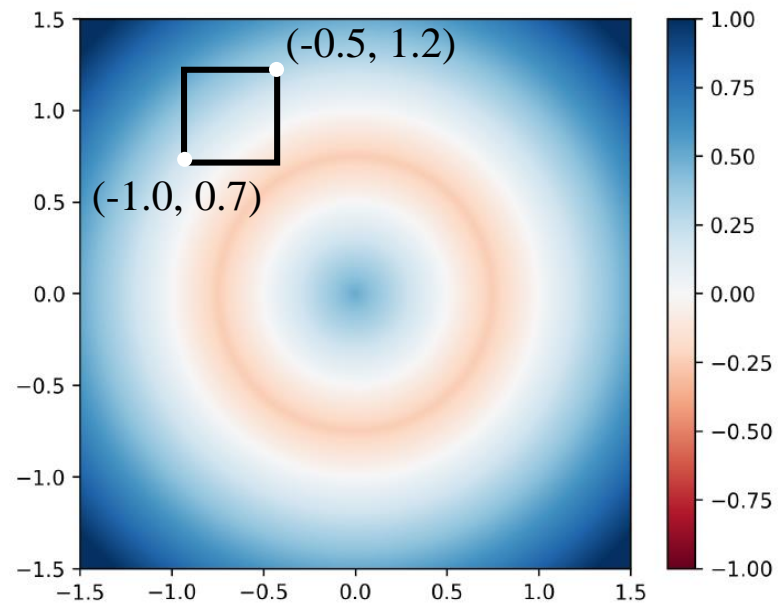
Instruction tape

Interpreter on the GPU

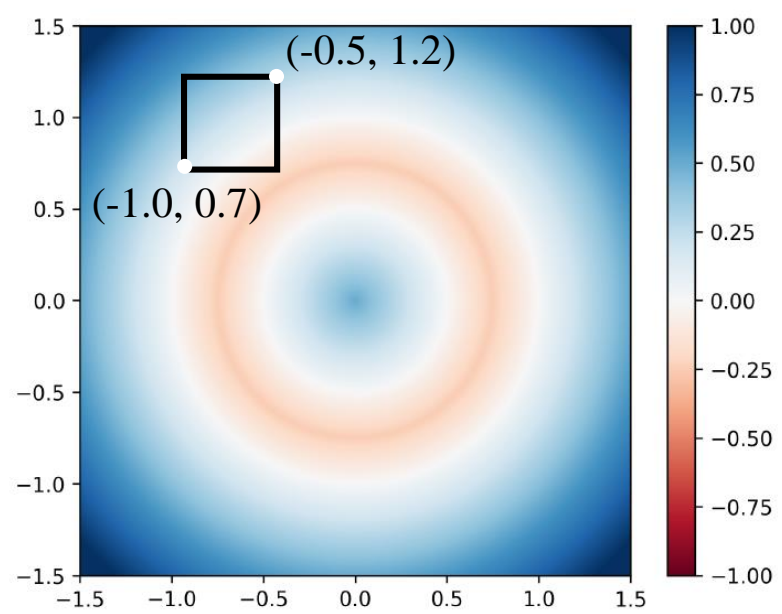
Key point: represent the implicit surface as a *tape*



Interval arithmetic

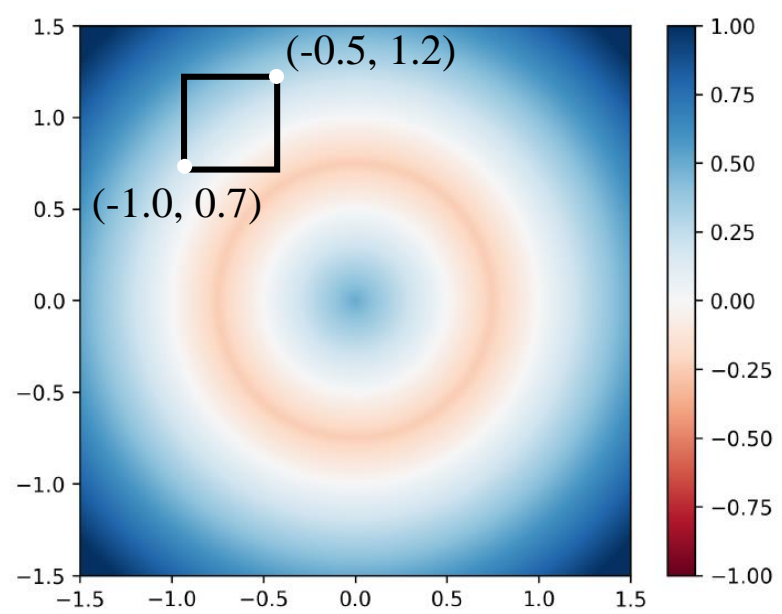


Interval arithmetic



opcode	lhs	rhs	out
X	—	—	$[-1.0, -0.5]$
Y	—	—	$[0.7, 1.2]$
SQUARE		—	
SQUARE		—	
ADD			
SQRT		—	
SUB		$1.0f$	
SUB	$0.5f$		
MAX			

Interval arithmetic

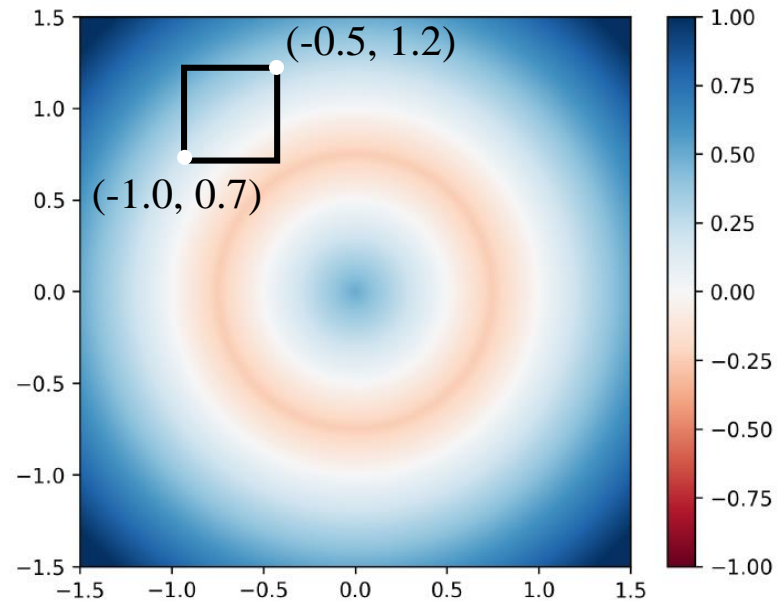


Square

opcode	lhs	rhs	out
X	—	—	$[-1.0, -0.5]$
Y	—	—	$[0.7, 1.2]$
SQUARE		—	
SQUARE		—	
ADD			
SQRT		—	
SUB		$1.0f$	
SUB	$0.5f$		
MAX			

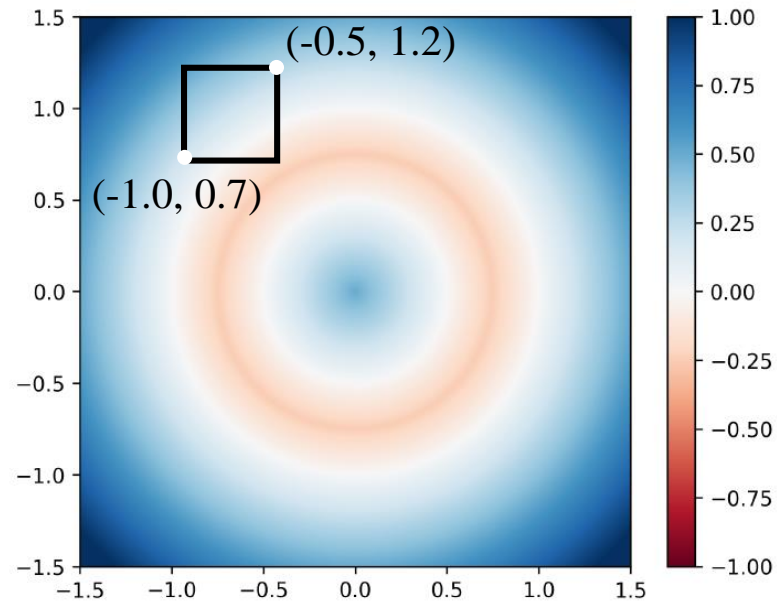
opcode	lhs	rhs	out
X	—	—	$[-1.0, -0.5]$
Y	—	—	$[0.7, 1.2]$
SQUARE	$[-1.0, -0.5]$	—	$[0.25, 1.0]$
SQUARE		—	
ADD			
SQRT		—	
SUB		$1.0f$	
SUB	$0.5f$		
MAX			

Interval arithmetic



opcode	lhs	rhs	out
X	—	—	$[-1.0, -0.5]$
Y	—	—	$[0.7, 1.2]$
SQUARE	$[-1.0, -0.5]$	—	$[0.25, 1.0]$
SQUARE	$[0.7, 1.2]$	—	$[0.49, 1.44]$
ADD	$[0.25, 1.0]$	$[0.49, 1.44]$	$[0.74, 2.44]$
SQRT	$[0.74, 2.44]$	—	$[0.86, 1.56]$
SUB	$[0.86, 1.56]$	1.0f	$[-0.14, 0.56]$
SUB	0.5f	$[0.86, 1.56]$	$[-1.06, -0.36]$
MAX	$[-0.14, 0.56]$	$[-1.06, -0.36]$	$[-0.14, 0.56]$

Interval arithmetic

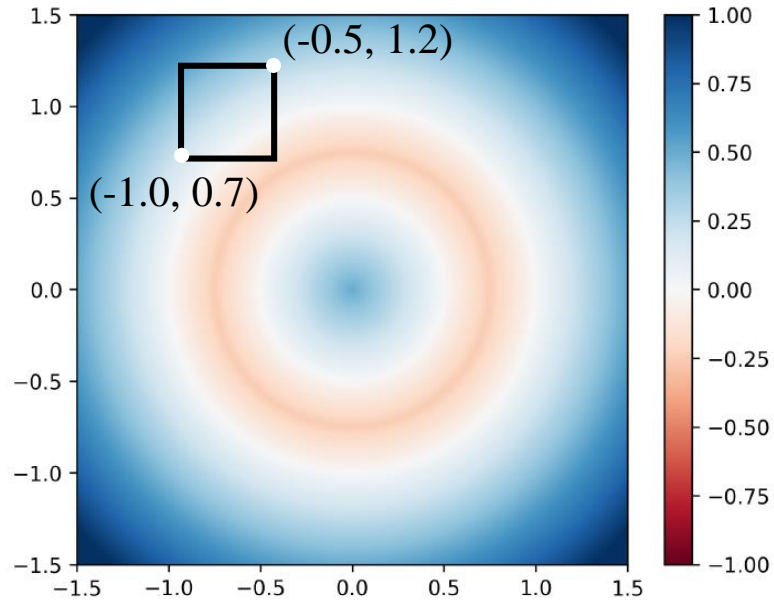


opcode	lhs	rhs	out
X	—	—	$[-1.0, -0.5]$
Y	—	—	$[0.7, 1.2]$
SQUARE	$[-1.0, -0.5]$	—	$[0.25, 1.0]$
SQUARE	$[0.7, 1.2]$	—	$[0.49, 1.44]$
ADD	$[0.25, 1.0]$	$[0.49, 1.44]$	$[0.74, 2.44]$
SQRT	$[0.74, 2.44]$	—	$[0.86, 1.56]$
SUB	$[0.86, 1.56]$	$1.0f$	$[-0.14, 0.56]$
SUB	$0.5f$	$[0.86, 1.56]$	$[-1.06, -0.36]$
MAX	$[-0.14, 0.56]$	$[-1.06, -0.36]$	$[-0.14, 0.56]$

Lower

Upper

Interval arithmetic



opcode	lhs	rhs	out
X	—	—	$[-1.0, -0.5]$
Y	—	—	$[0.7, 1.2]$
SQUARE	$[-1.0, -0.5]$	—	$[0.25, 1.0]$
SQUARE	$[0.7, 1.2]$	—	$[0.49, 1.44]$
ADD	$[0.25, 1.0]$	$[0.49, 1.44]$	$[0.74, 2.44]$
SQRT	$[0.74, 2.44]$	—	$[0.86, 1.56]$
SUB	$[0.86, 1.56]$	$1.0f$	$[-0.14, 0.56]$
SUB	$0.5f$	$[0.86, 1.56]$	$[-1.06, -0.36]$
MAX	$[-0.14, 0.56]$	$[-1.06, -0.36]$	$[-0.14, 0.56]$

Lower Upper

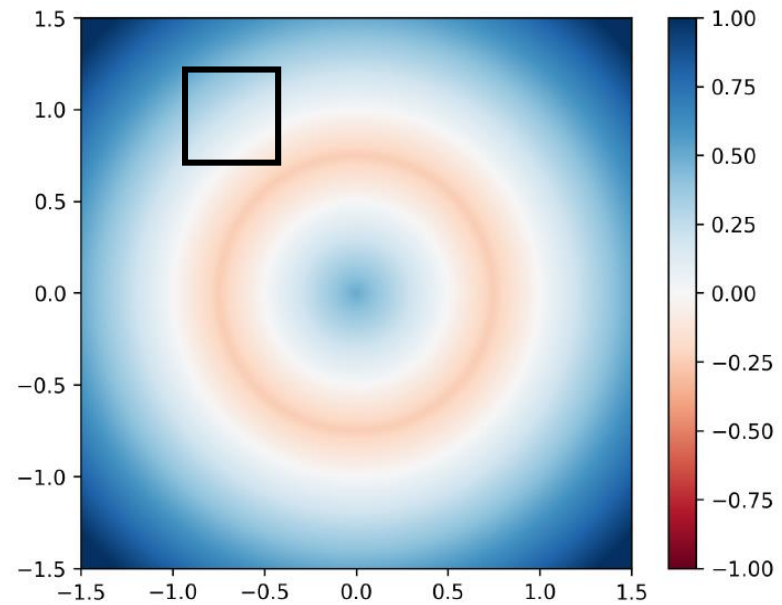
Returns the lower and upper values found in the interval

if $lower > 0$ ➔ Empty interval

if $upper < 0$ ➔ Filled interval

otherwise ➔ Ambiguous

Tape simplification



opcode	lhs	rhs	out
X	—	—	$[-1.0, -0.5]$
Y	—	—	$[0.7, 1.2]$
SQUARE	$[-1.0, -0.5]$	—	$[0.25, 1.0]$
SQUARE	$[0.7, 1.2]$	—	$[0.49, 1.44]$
ADD	$[0.25, 1.0]$	$[0.49, 1.44]$	$[0.74, 2.44]$
SQRT	$[0.74, 2.44]$	—	$[0.86, 1.56]$
SUB	$[0.86, 1.56]$	$1.0f$	$[-0.14, 0.56]$
SUB	$0.5f$	$[0.86, 1.56]$	$[-1.06, -0.36]$
MAX	$[-0.14, 0.56]$	$[-1.06, -0.36]$	$[-0.14, 0.56]$

Tape simplification

opcode	lhs	rhs	out
X	–	–	$[-1.0, -0.5]$
Y	–	–	$[0.7, 1.2]$
SQUARE	$[-1.0, -0.5]$	–	$[0.25, 1.0]$
SQUARE	$[0.7, 1.2]$	–	$[0.49, 1.44]$
ADD	$[0.25, 1.0]$	$[0.49, 1.44]$	$[0.74, 2.44]$
SQRT	$[0.74, 2.44]$	–	$[0.86, 1.56]$
SUB	$[0.86, 1.56]$	$1.0f$	$[-0.14, 0.56]$
SUB	$0.5f$	$[0.86, 1.56]$	$[-1.06, -0.36]$
MAX	$[-0.14, 0.56]$	$[-1.06, -0.36]$	$[-0.14, 0.56]$

Tape simplification

opcode	lhs	rhs	out
X	–	–	$[-1.0, -0.5]$
Y	–	–	$[0.7, 1.2]$
SQUARE	$[-1.0, -0.5]$	–	$[0.25, 1.0]$
SQUARE	$[0.7, 1.2]$	–	$[0.49, 1.44]$
ADD	$[0.25, 1.0]$	$[0.49, 1.44]$	$[0.74, 2.44]$
SQRT	$[0.74, 2.44]$	–	$[0.86, 1.56]$
SUB	$[0.86, 1.56]$	$1.0f$	$[-0.14, 0.56]$
SUB	$0.5f$	$[0.86, 1.56]$	$[-1.06, -0.36]$
MAX	$[-0.14, 0.56]$	$[-1.06, -0.36]$	$[-0.14, 0.56]$

Tape simplification

Use interval results at min/max clauses to detect inactive clauses in a particular region

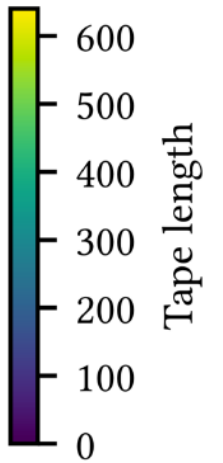
Construct shortened tape

That tape is valid for all sub regions contained within the parent region

Tape simplification: effectiveness

But this rough magic I
here abjure, and when
I have required some
heavenly music, which even
now I do, to work mine
end upon their senses that
this airy charm is for, I'll
break my staff, bury it
certain fathoms in the
earth, and deeper than did
ever plummet sound
I'll drown my book.

Original: 6056 clauses

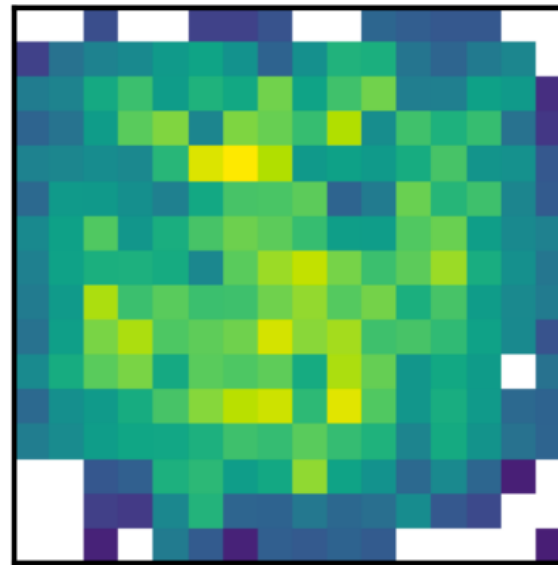


Tape simplification: effectiveness

White is completely inside/outside

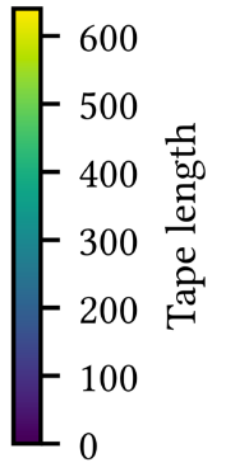
But this rough magic I
here abjure, and when
I have required some
heavenly music, which even
now I do, to work mine
end upon their senses that
this airy charm is for, I'll
break my staff, bury it
certain fathoms in the
earth, and deeper than did
ever plummet sound
I'll drown my book.

Original: 6056 clauses



356 ± 125 clauses

First pass (64x64 tiles)



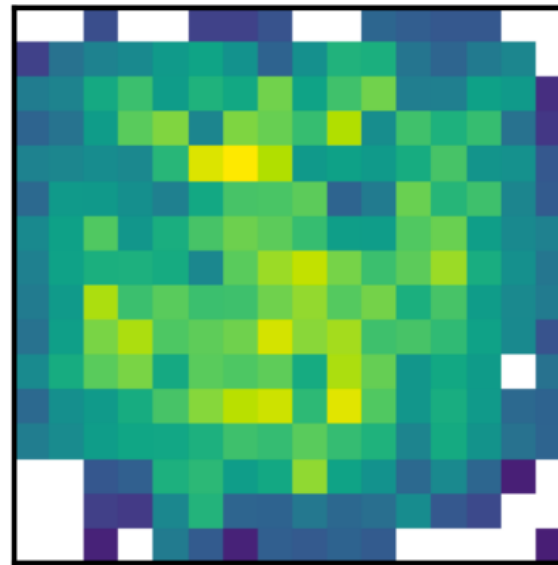
Tape simplification: effectiveness

White is completely inside/outside



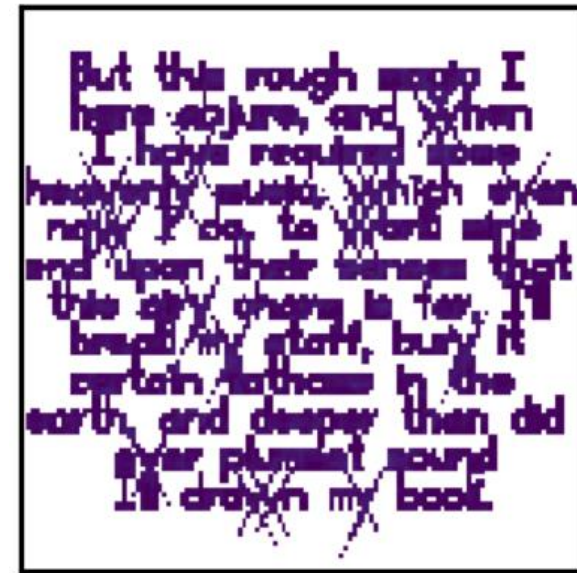
But this rough magic I
here abjure, and when
I have required some
heavenly music, which even
now I do, to work mine
end upon their senses that
this airy charm is for, I'll
break my staff, bury it
certain fathoms in the
earth, and deeper than did
ever plummet sound
I'll drown my book.

Original: 6056 clauses



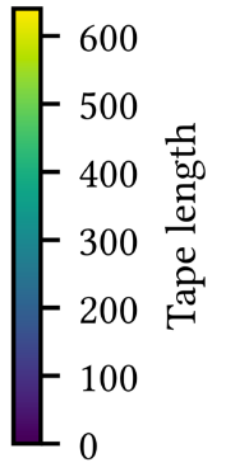
356 ± 125 clauses

First pass (64x64 tiles)



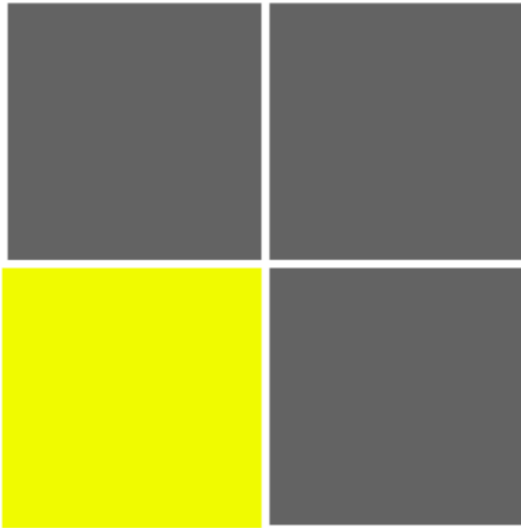
28 ± 13 clauses

Second pass (8x8 subtiles)

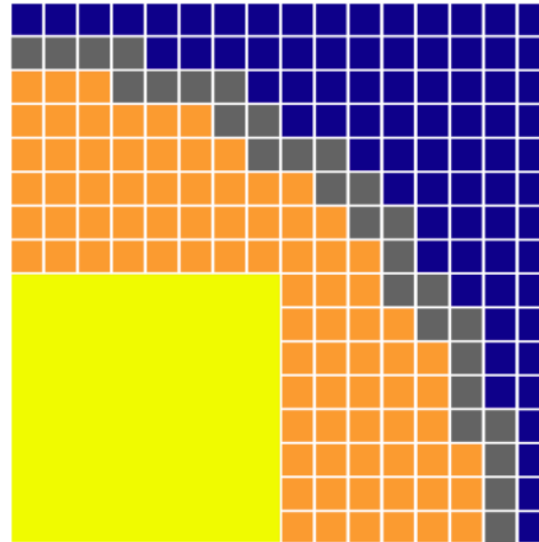


Full pipeline

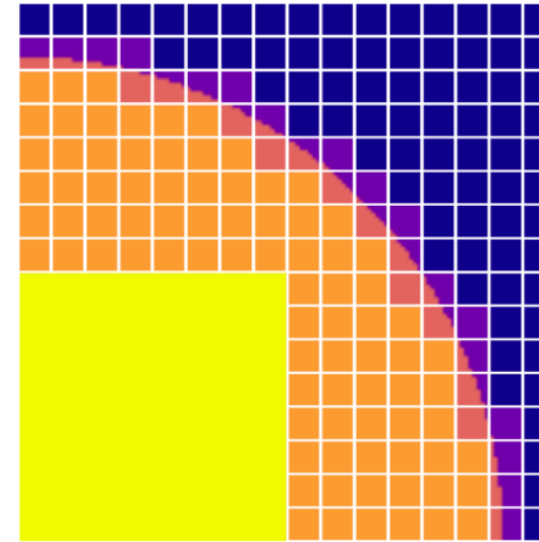
First pass



Second pass



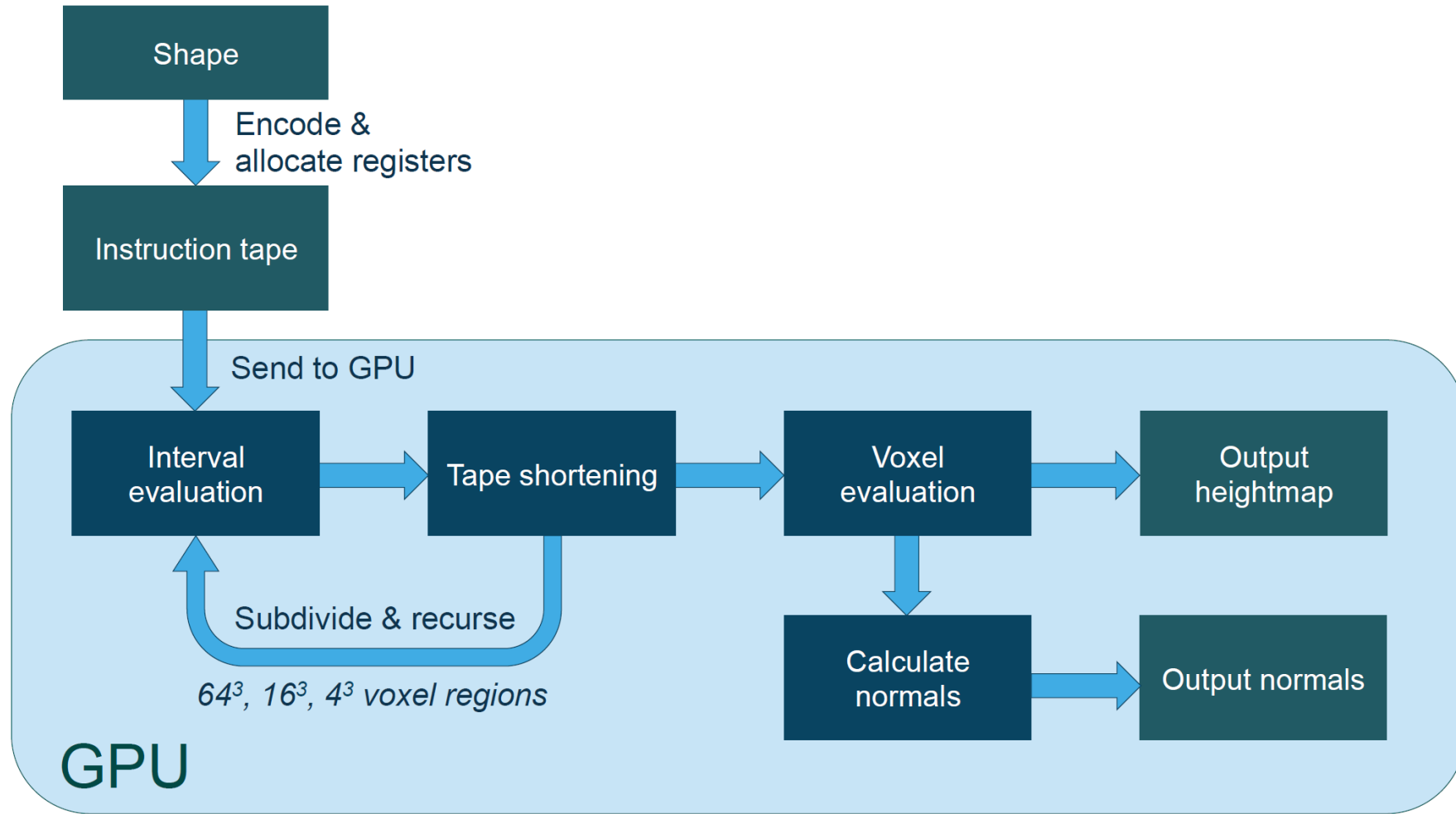
Per-pixel evaluation



-  64x64 filled tile
-  8x8 filled tile
-  8x8 empty tile
-  8x8 ambiguous tile

2D pipeline

3D Full pipeline



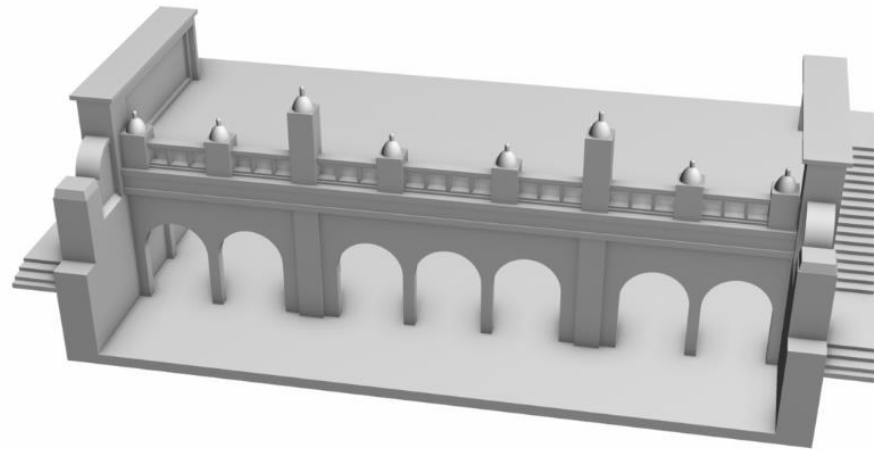
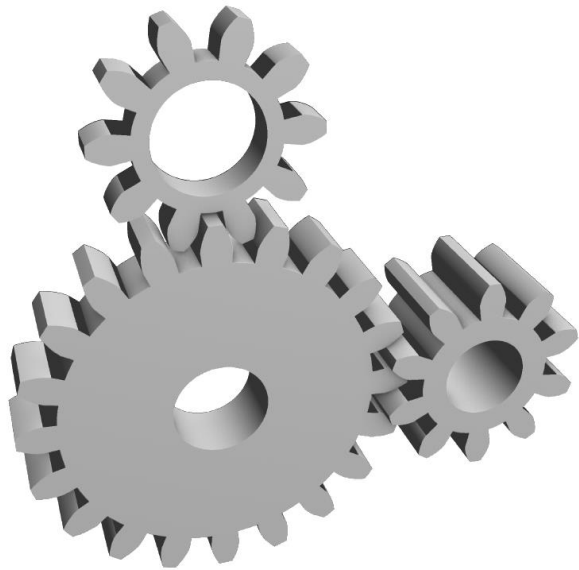
Full pipeline

- Interval evaluation of 64x64x64-voxel **tiles**
 - Split ambiguous tiles into 64 **subtiles** (16x16x16-voxels) each
- Interval evaluation of **subtiles**
 - Split ambiguous subtiles into 64 **microtiles** (4x4x4-voxels) each
- Interval evaluation of **microtiles**
- Per-**voxel** evaluations of remaining ambiguous **microtiles**
- Per-**pixel** evaluation, using automatic differentiation to find normals
- *Optional*: post-processing depth + normal buffer to generate final image

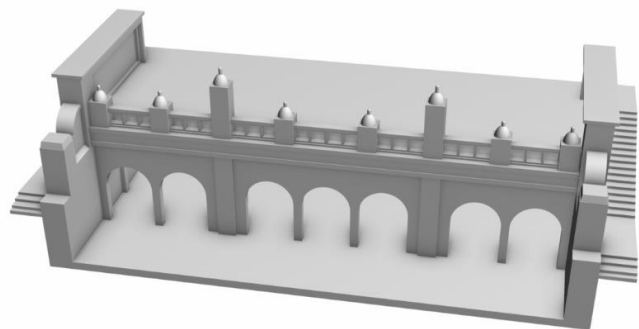
Results



Model	Clauses	CSG	Dimensions
Architectural model	961	465	3D
Bear head sculpt	541	27	3D
Text benchmark	6056	2354	2D
Gears	1735	374	Both



Results



Architectural model

Size	Frame time (ms)		
	GeForce GT 750M	GTX 1080 Ti	Tesla V100
256 ³	34.3	5.5	3.2
512 ³	73.9	9.9	5.3
1024 ³	189.9	22.6	12.2
1536 ³	331.9	39.3	20.8
2048 ³	510.7	60.6	31.9

Gears (3D)

Size	Frame time (ms)		
	GeForce GT 750M	GTX 1080 Ti	Tesla V100
256 ³	65.0	9.4	6.2
512 ³	154.5	16.6	9.2
1024 ³	426.2	40.3	23.1
1536 ³	930.3	72.0	39.5
2048 ³	—	115.4	62.0

Bear head sculpt

Size	Frame time (ms)		
	GeForce GT 750M	GTX 1080 Ti	Tesla V100
256 ³	111.3	11.3	5.2
512 ³	503.6	41.1	20.3
1024 ³	2352.1	191.0	88.3
1536 ³	—	504.2	228.3
2048 ³	—	1053.2	437.3

Discussion

Works in both 2D and 3D

Requires only C^0 continuity

Interactive framerates, even for complex models

Scales with GPU power

Works best with classic CSG operators (min/max)

Lots of GPU memory required

Lack of examples: only three scenes

Complex GPU implementation

Open source, in cuda, macOS only: <https://github.com/mkeeter/mpr>

```
/*
Reference implementation for
"Massively Parallel Rendering of Complex Closed-Form Implicit Surfaces"
(SIGGRAPH 2020)

This Source Code Form is subject to the terms of the Mozilla Public
License, v. 2.0. If a copy of the MPL was not distributed with this file,
You can obtain one at http://mozilla.org/MPL/2.0/.

Copyright (C) 2019-2020 Matt Keeter
*/
#pragma once

// A clause is defined as a 64-bit value.
//
// These macros let us address individual parts of that value, in a way
// which almost definitely is Undefined Behavior.
#define OP(d) (((uint8_t*)(d))[0])
#define I_OUT(d) (((uint8_t*)(d))[1])
#define I_LHS(d) (((uint8_t*)(d))[2])
#define I_RHS(d) (((uint8_t*)(d))[3])
#define IMM(d) (((float*)(d))[1])
#define JUMP_TARGET(d) (((int32_t*)(d))[1])
```

<https://github.com/mkeeter/mpr>

References

Paper page: <https://www.mattkeeter.com/research/mpr/>

Paper accompanying blog post: <https://www.mattkeeter.com/projects/siggraph/>

Author website: <https://www.mattkeeter.com/>

Interval arithmetic paper: http://fab.cba.mit.edu/classes/S62.12/docs/Duff_interval_CSG.pdf