A Geometry Engine from First Principles

Three years into a six month project

Julia Longtin

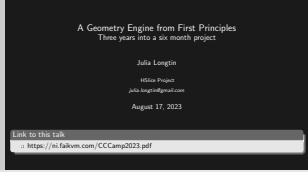
HSlice Project

julia.longtin@gmail.com

August 17, 2023

Link to this talk

- https://ni.faikvm.com/CCCamp2023.pdf



Who am I? (Role)

- Free Software Developer
  - Maintainer of ImplicitCAD
  - Developer of HSlice
- Work at Wire.com
  - The presented work has no affiliation with my employer

Who am I? (Role)

- Free Software Developer
  - Maintainer of ImplicitCAD
  - Developer of HSlice
- Work at Wire.com
  - The presented work has no affiliation with my employer

Implicitcad is a 3d modelling system, and HSlice is the program i've been working on, that makes those models printable.Wire is an end-to-end encrypted communications platform. think: something like slack meets signal, but open source everything.

Who am I? (???)

- From Washington, DC.. and Northwest Arkansas.
- Have lived in Berlin with my wife for 4+ years.
- Transhumanist, Haskeller, CyberPunk enthusiast.
- Created a method for converting 3D prints to aluminium using microwaves (see: 31c3).

Who am I? (???)

- From Washington, DC.. and Northwest Arkansas.
- Have lived in Berlin with my wife for 4+ years.
- Transhumanist, Haskeller, CyberPunk enthusiast.
- Created a method for converting 3D prints to aluminium using microwaves (see: 31c3).

I am from a very disadvantaged part of the united states, so have seen a lot of suffering which partially drives my workCyberPunk enthusiast is a polite way of saying I watch too much ghost in the shell, and spend too much time imagining how more technology could help both myself, and the people I care about.which is what gave me the confidence to begin this project, and estimate it at 6 months.

Why am I here?

I'm making a new kind of slicer, for 3D printing!

- Hope:
  - Hoping improving slicer technology can help non-hackers.
- Inspiration:
  - Glimmers of non-planar slicing in the 3D printing world.
- Results I'm looking for (Better Slicers!):
  - Parallelizing un-parallelized slicing algorithms.
  - Increasing slicer precision.
  - Increasing speed.

Why am I here?

I'm making a new kind of slicer, for 3D printing!
- Hope:
  - Hoping improving slicer technology can help non-hackers.
- Inspiration:
  - Glimmers of non-planar slicing in the 3D printing world.
- Results I'm looking for (Better Slicers!):
  - Parallelizing un-parallelized slicing algorithms.
  - Increasing slicer precision.
  - Increasing speed.

increased precision and increased speed is critical as we try to tackle more detailed prints made of more materials.

What did I have before starting?

Where I Began
What did I have before starting?

Status Quo

My 3d printing workflow:

- Open up a text editor, and describe my object.
- Hand it to my 3d modelling system.
- Wait.
- Receive an STL file. Open it in a visualizer, go back to step 1 if its wrong.
- Open up a slicer, and hand it my STL file.
- Pick the material I want to use, and the quality level I want in the print.
- Wait.
- Receive a GCode file.
- Stick this file on a memory stick, and plug it into my printer. hit 'Print'.
- Wait.
- Receive a thing.

Status Quo
My 3d printing workflow:
- Open up a text editor, and describe my object.
- Hand it to my 3d modelling system.
- Wait.
- Receive an STL file. Open it in a visualizer, go back to step 1 if its wrong.
- Open up a slicer, and hand it my STL file.
- Pick the material I want to use, and the quality level I want in the print.
- Wait.
- Receive a GCode file.
- Stick this file on a memory stick, and plug it into my printer. hit 'Print'.
- Wait.
- Receive a thing.

I once waited 3 hours to slice a shoe in a single material.

What is ImplicitCAD?

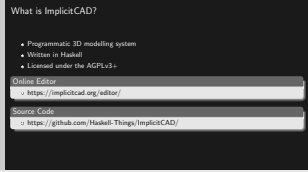
- Programmatic 3D modelling system
- Written in Haskell
- Licensed under the AGPLv3+

Online Editor

- https://implicitcad.org/editor/

Source Code

- https://github.com/Haskell-Things/ImplicitCAD/

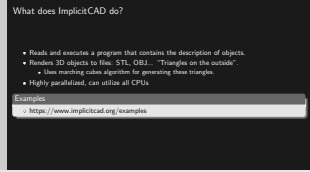


What does ImplicitCAD do?

- Reads and executes a program that contains the description of objects.
- Renders 3D objects to files: STL, OBJ... "Triangles on the outside".
  - Uses marching cubes algorithm for generating these triangles.
- Highly parallelized, can utilize all CPUs

Examples

- https://www.implicitcad.org/examples



If you've used openscad, implicitcad's language is a subset of that language

What is a slicer's job?

We're going to use a basic definition.
A slicer:

- Reads a file containing one or more objects.
  - Objects are given as a set of triangles on the exterior of where the object begins.
- Divides objects into layers.
- Computes instructions to fill in those layers. Different layer heights give different quality of print.
- Computes instructions to manage the non-geometry parts of 3D printing(temperature, speeds of movement, fan control..).
- Writes a GCode file contaiting these instructions, which can be given to a printer to produce the object(s) from the file.

Note: We're discussing FDM(Fused Deposition of Material) printing during this talk, but many other methods exist, each with their own plusses and minuses.

What is a slicer's job?

We're going to use a basic definition.
A slicer:
- Reads a file containing one or more objects.
  - Objects are given as a set of triangles on the exterior of where the object begins.
- Divides objects into layers.
- Computes instructions to fill in those layers. Different layer heights give different quality of print.
- Computes instructions to manage the non-geometry parts of 3D printing(temperature, speeds of movement, fan control..).
- Writes a GCode file contaiting these instructions, which can be given to a printer to produce the object(s) from the file.

Note: We're discussing FDM(Fused Deposition of Material) printing during this talk, but many other methods exist, each with their own plusses and minuses.

a pyramid could be represented as four triangles for what you see above the surface, and two in the base, making a square.GCode is basically instructions for "go from point A to point B at speed X while extruding material at Y volume per second"FDM... think, "a hot glue gun on a robot".

Why do we need better slicers?

- Performance and Scalability(number of CPUs):
  - Slicers commonly use external geometry engines and algorithms that are single threaded..
  - .. and are written in imperative languages.
  - .. which hinders using a GPU, FPGA, or other type of accelerator.
- Precision:
  - Current major slicers use double precision floating point and linear algebra.
  - Projective Geometry and more advanced algorithms can make use of the same double precision floating point, while introducing less error.
- Scalability of printing:
  - Models are getting more and more complex. We are learning to slice aross planes, to print with more materials at once, and are producing smaller objects (bioprinting, anyone?).

Why do we need better slicers?
- Performance and Scalability(number of CPUs):
  - Slicers commonly use external geometry engines and algorithms that are single threaded..
  - .. and are written in imperative languages.
  - .. which hinders using a GPU, FPGA, or other type of accelerator.
- Precision:
  - Current major slicers use double precision floating point and linear algebra.
  - Projective Geometry and more advanced algorithms can make use of the same double precision floating point, while introducing less error.
- Scalability of printing:
  - Models are getting more and more complex. We are learning to slice aross planes, to print with more materials at once, and are producing smaller objects (bioprinting, anyone?).

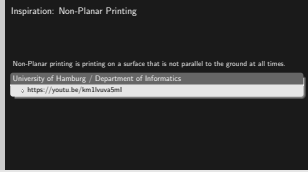
If you're wondering what imperative programming is, if you're giving your computer a sequence of instructions, that's imperative. if you're giving it a list of formulas, that's functional programming.

Inspiration: Non-Planar Printing

Non-Planar printing is printing on a surface that is not parallel to the ground at all times.

University of Hamburg / Department of Informatics

- https://youtu.be/km1lvuva5ml



normally, we stack layers onto a 3D print like we were stacking strangely shaped pancakes. non-planar slicing breaks this pattern.

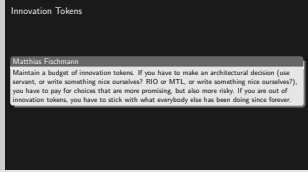
What did I think I would do differently?

Innovation Tokens
What did I think I would do differently?

Innovation Tokens

Matthias Fischmann

Maintain a budget of innovation tokens. If you have to make an architectural decision (use servant, or write something nice ourselves? RIO or MTL, or write something nice ourselves?), you have to pay for choices that are more promising, but also more risky. If you are out of innovation tokens, you have to stick with what everybody else has been doing since forever.



Innovation Token 1: Using Haskell

- A Functional programming language.
- Lazy:
  - Does not perform an operation until its needed.
- Separates Pure from Impure code:
  - Pure code does not retain state between function calls.
  - Pure code is simpler to parallelize, simpler to test..
  - .. and possible to port to FPGAs / GPUs.

Innovation Token 1: Using Haskell
- A Functional programming language.
- Lazy:
  - Does not perform an operation until its needed.
- Separates Pure from Impure code:
  - Pure code does not retain state between function calls.
  - Pure code is simpler to parallelize, simpler to test..
  - .. and possible to port to FPGAs / GPUs.

Coding in pure Haskell is more like writing math. you don't give your compiler a sequence of instructions, you give a big formula, describing the relationship between your problems. this gives the compiler more opportunities to try to out-think you, and to optimize your code.

Innovation Token 2: Building my own geometry engine

- Not using the giant geometry libraries from the 80s.
- Allows the use of more advanced algorithms and algebras..
- Avoided interfacing with non-Haskell code:
  - Allows for knowing what parts of the library (all of them!) are Pure code.
  - Allows easier parallelization, refactoring.

Innovation Token 2: Building my own geometry engine

- Not using the giant geometry libraries from the 80s.
- Allows the use of more advanced algorithms and algebras..
- Avoided interfacing with non-Haskell code:
  - Allows for knowing what parts of the library (all of them!) are Pure code.
  - Allows easier parallelization, refactoring.

I implemented this progressively, writing the parts I needed, as I needed them.

Innovation Token 3: Using Projective Geometric Algebra

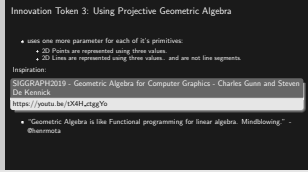
- uses one more parameter for each of it's primitives:
  - 2D Points are represented using three values.
  - 2D Lines are represented using three values.. and are not line segments.

Inspiration:

SIGGRAPH2019 - Geometric Algebra for Computer Graphics - Charles Gunn and Steven De Kennick

https://youtu.be/tX4H_ctggYo

- "Geometric Algebra is like Functional programming for linear algebra. Mindblowing." - @henrmota



The authors of this work don't deliniate well between Projective Geometric Algebra, and Geometric algebra. in fact they seem to be coming up with some other term now, so the language around this is more confusing than the math. we're using PGA to describe the primitives and functions on those primitives, and GA for the operators.We are using a clifford algebra of (2,0,1)*

Innovation Token 4: New Geometry Algorithms

- Faster algorithms available.
- Parallelism? I think so...

Innovation Token 4: New Geometry Algorithms

- Faster algorithms available.
- Parallelism? I think so...

How did I get started?

Beginning Work
How did I get started?

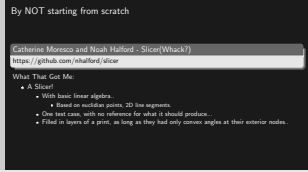
By NOT starting from scratch

Catherine Moresco and Noah Halford - Slicer(Whack?)

https://github.com/nhalford/slicer

What That Got Me:

- A Slicer!
  - With basic linear algebra..
    - Based on euclidian points, 2D line segments.
  - One test case, with no reference for what it should produce...
  - Filled in layers of a print, as long as they had only convex angles at their exterior nodes..



the first mistake I think most developers make is assuming no-one has tried what you're trying to do.our definition of convex: If you put a baloon around it, the baloon would touch all of the points where two line segments meet.

How did it go?

Spending our Innovation Tokens

How did it go?

Experience of only using Haskell

As I have maintained ImplicitCAD for many years, Haskell generally got “out of my way”, with the following exceptions:

- Floating point rounding control is a separate library, which seems un-loved, and does not work on newer versions of the haskell compiler yet.
- There is always a better way to learn to do things in Haskell, which led to me having many moments of “who wrote this crap?” as I re-factored over the years.
- The compiler and type system worked closely with me, when refactoring! :)
- When implementing a function where I didn’t know how to handle all of the possible cases, I could set up a filter, and call error() in the cases I don’t know how to implement yet.

Experience of only using Haskell

As I have maintained ImplicitCAD for many years, Haskell generally got “out of my way”, with the following exceptions:

- Floating point rounding control is a separate library, which seems un-loved, and does not work on newer versions of the haskell compiler yet.
- There is always a better way to learn to do things in Haskell, which led to me having many moments of “who wrote this crap?” as I re-factored over the years.
- The compiler and type system worked closely with me, when refactoring! :)
- When implementing a function where I didn’t know how to handle all of the possible cases, I could set up a filter, and call error() in the cases I don’t know how to implement yet.

Experience of adding Geometric Algebra

We needed a Geometric Algebra library, to do the math for Projective Geometric Algebra functions.

A Geometric Algebra is a system of algebra operating on values in vector spaces.

Think: Number plus axis, where we have a lot of axes, and combinations of axes.

- The papers I was using did not define in detail the operators they were using, which slowed development. The word 'multiply' had a lot of meanings.
  - It took quite some time to determine how to perform the operations described.
  - Then I had to learn how to abuse sort() for performing reduction of geometric products.
  - Finally, I learned the hard way that i needed to treat scalars as their own axis in the geometric product.
  - .. I still don't have the math for the origin working right. avoid (0,0). :)

Experience of adding Geometric Algebra

We needed a Geometric Algebra library, to do the math for Projective Geometric Algebra functions.
A Geometric Algebra is a system of algebra operating on values in vector spaces.
Think: Number plus axis, where we have a lot of axes, and combinations of axes.
- The papers I was using did not define in detail the operators they were using, which slowed development. The word 'multiply' had a lot of meanings.
  - It took quite some time to determine how to perform the operations described.
  - Then I had to learn how to abuse sort() for performing reduction of geometric products.
  - Finally, I learned the hard way that i needed to treat scalars as their own axis in the geometric product
  - . I still don't have the math for the origin working right. avoid (0,0) .:)

I am no mathematiciansomehow, I ended up with three operators, which when used together, do the work of the two sub operators they defined (wedge and dot product).

Resources to prevent the same mistakes

BiVector.net: 2D Projective Geometric Algebra Cheat Sheet

https://bivector.net/2DPGA.pdf

My Code! :)

github.com/Haskell-Things/HSlice/blob/master/Graphics/Slicer/Math/GeometricAlgebra.hs

Resources to prevent the same mistakes
BiVector.net: 2D Projective Geometric Algebra Cheat Sheet
https://bivector.net/2DPGA.pdf
My Code! :)
github.com/Haskell-Things/HSlice/blob/master/Graphics/Slicer/Math/GeometricAlgebra.hs

Experience of adding Projective Geometric Algebra

Long and painful. Arguably incomplete in the error estimation area.

- There are many papers by Charles Gunn, which assume a level of math I'm not at.
  - The symbols used in many places are not defined.
  - Normalization and canonicalization were new concepts to me. "you mean, after some operations, you have to perform a specific operation on the result so other operations work correctly?"
  - Normalization and canonicalization are necesary.. sometimes. They are often skipped in the papers. It took much trial and error to find the right places to perform each operation.
- Using the type system of haskell to only normalize and canonicalize when necessary was necessary so as to not accidentaly throw away precision and performance.
- Finding symmetry breaking operations took work. Some of the operations you need when reasoning about geometry were not well documented.
  Think: "if I compare line A to line B, does line A point toward the left/right/forward/backward compared to line B?"

Experience of adding Projective Geometric Algebra

Long and painful. Arguably incomplete in the error estimation area.

- There are many papers by Charles Gunn, which assume a level of math I'm not at.
  - The symbols used in many places are not defined.
  - Normalization and canonicalization were new concepts to me. "you mean, after some operations, you have to perform a specific operation on the result so other operations work correctly?"
  - Normalization and canonicalization are necesary. sometimes. They are often skipped in the papers. It took much trial and error to find the right places to perform each operation.
- Using the type system of haskell to only normalize and canonicalize when necessary was necessary so as to not accidentaly throw away precision and performance.
- Finding symmetry breaking operations took work. Some of the operations you need when reasoning about geometry were not well documented.
  Think: "if I compare line A to line B, does line A point toward the left/right/forward/backward compared to line B?"

Resources to prevent the same mistakes

Charles G. Gunn: Projective Geometric Algebra

https://arxiv.org/pdf/1901.05873.pdf

BiVector forum Projective GA category

https://discourse.bivector.net/c/projective-ga/

My Code! :)

github.com/Haskell-Things/HSlice/blob/master/Graphics/Slicer/Math/PGAPrimitives.hs

Resources to prevent the same mistakes
Charles G. Gunn: Projective Geometric Algebra
https://arxiv.org/pdf/1901.05873.pdf
BiVector forum Projective GA category
https://discourse.bivector.net/c/projective-ga/
My Code! :)
github.com/Haskell-Things/HSlice/blob/master/Graphics/Slicer/Math/PGAPrimitives.hs

Experience of adding new geometry algorithms

Longer, and painfuller. Had to learn about a part of computational geometry I hadn't even heard of before: Straight Skeleton Generation.
Implementation is incomplete. Just starting to slice complicated shapes, no hole handling yet.

- Straight Skeletons are deceptively simple. I estimated difficulty without even understanding this part.
- Many papers have been written on the subject, but very few have implementations.. and none have Haskell implementations.
- The mathematical approach of measuring algorithmic speed does not fit well with our modern multi-core world.
- I found one paper that's approachable.. but still have not implemented it completely.

Experience of adding new geometry algorithms

Longer, and painfuller. Had to learn about a part of computational geometry I hadn't even heard of before: Straight Skeleton Generation.
Implementation is incomplete. Just starting to slice complicated shapes, no hole handling yet.

- Straight Skeletons are deceptively simple. I estimated difficulty without even understanding this part.
- Many papers have been written on the subject, but very few have implementations.. and none have Haskell implementations.
- The mathematical approach of measuring algorithmic speed does not fit well with our modern multi-core world.
- I found one paper that's approachable.. but still have not implemented it completely.

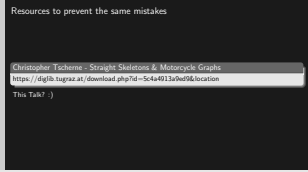
If it is less steps, but requires a single fast core.. it will run horribly on modern computers.

Resources to prevent the same mistakes

Christopher Tscherne - Straight Skeletons & Motorcycle Graphs

https://diglib.tugraz.at/download.php?id=5c4a4913a9ed9&location

This Talk? :)



When I went looking for other slicers implementing these functions, i found a majority of attempts to write slicers died at this step. Reach out if you have trouble.

What did that get us?

Results
What did that get us?

HSlice!

A Haskell based slicer!

- A Geometric Algebra library!
  - With Floating Point error tracking.
- A mixed Euclidian / Projective geometry engine.
  - With a large (260 at present) test suite.
- A Parallel Straight Skeleton solver.
- A binary styled after the Cura slicing engine.
- Just a hair more precision..

HSlice!

A Haskell based slicer!
- A Geometric Algebra library!
  - With Floating Point error tracking.
- A mixed Euclidian / Projective geometry engine.
  - With a large (260 at present) test suite.
- A Parallel Straight Skeleton solver.
- A binary styled after the Cura slicing engine.
- Just a hair more precision..

naming is hard, so i didn't try.if i slice with exactly the same settings and features between my slicer and cura, i save just a touch of plastic, because the lines are straighter, and less floating point error makes it into the print.

So what is a Straight Skeleton, and why do we need to solve it?

Results
So what is a Straight Skeleton, and why do we need to solve it?

Results

The results of this work break down into three categories:

- Slicing parallelism improvements
- Floating point error tracking
- Property testing geometry

Results

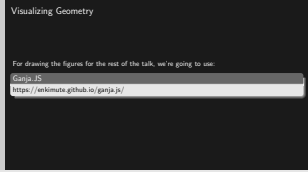
The results of this work break down into three categories:
- Slicing parallelism improvements
- Floating point error tracking
- Property testing geometry

Visualizing Geometry

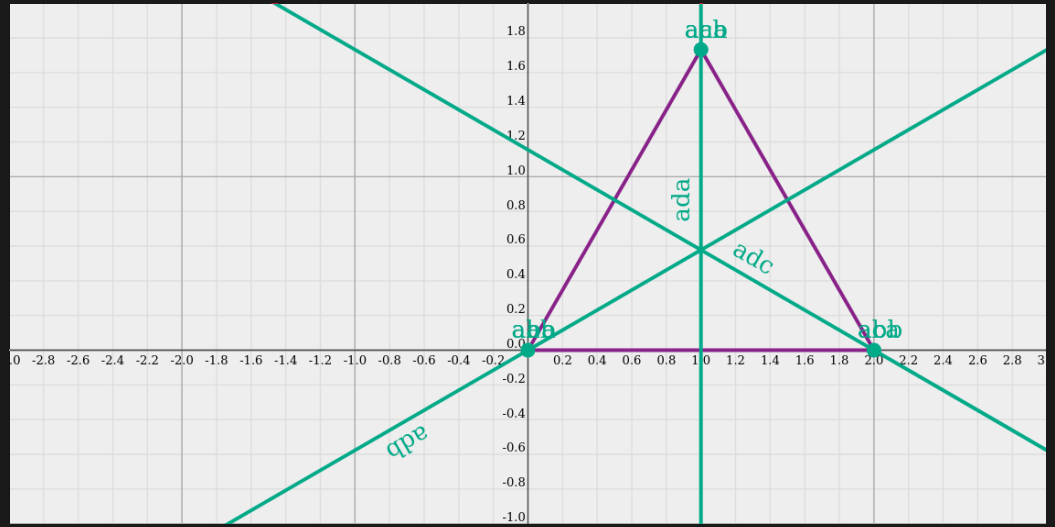
For drawing the figures for the rest of the talk, we're going to use:

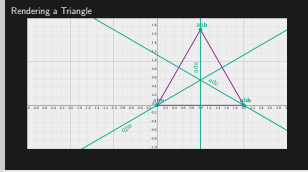
Ganja.JS

https://enkimute.github.io/ganja.js/



Rendering a Triangle





Point out the exterior nodes, and the interior node.weaknesses of ganja.js: always focuses on the origin, no ray primitive, labels for two points at the same point overwrite each other.

Slicing Primitives: Contour

- A sequence of at least three line segments.
- Closed.
- Only two line segments end/start at any point.
- No line segments intersect any other line segment, except at their end/start point.
- Interesting Attributes:
  - Has a winding, clockwise or counterclockwise.
- Interesting Operations:
  - outset: to draw a bigger form of the contour where each line segment is a given distance from its companion in the original contour.
  - inset: to draw a smaller form of the contour where each line segment is a given distance from its companion in the original contour.

Slicing Primitives: Contour

- A sequence of at least three line segments.
- Closed.
- Only two line segments end/start at any point.
- No line segments intersect any other line segment, except at their end/start point.
- Interesting Attributes:
  - Has a winding, clockwise or counterclockwise.
- Interesting Operations:
  - outset: to draw a bigger form of the contour where each line segment is a given distance from its companion in the original contour.
  - inset: to draw a smaller form of the contour where each line segment is a given distance from its companion in the original contour.

insetting and outsetting are critically required when you're trying to print the edge of an object. if you use a sharpie, and draw a square according to instructions, you have actually drawn too big of a square. you need to inset by half of the width of the marker you're using. something similar is required when 3d printing, so you don't produce over-sized objects.

Slicing Primitives: Exterior Nodes

- Exist at the point two line segments intersect forming a concave angle from the perspective of the inside of the contour.
- Always emit a ray toward the inside of the contour, along the bisector of the angle created by the two line segments.

Slicing Primitives: Exterior Nodes

- Exist at the point two line segments intersect forming a concave angle from the perspective of the inside of the contour.
- Always emit a ray toward the inside of the contour, along the bisector of the angle created by the two line segments.

Slicing Primitives: Interior Nodes

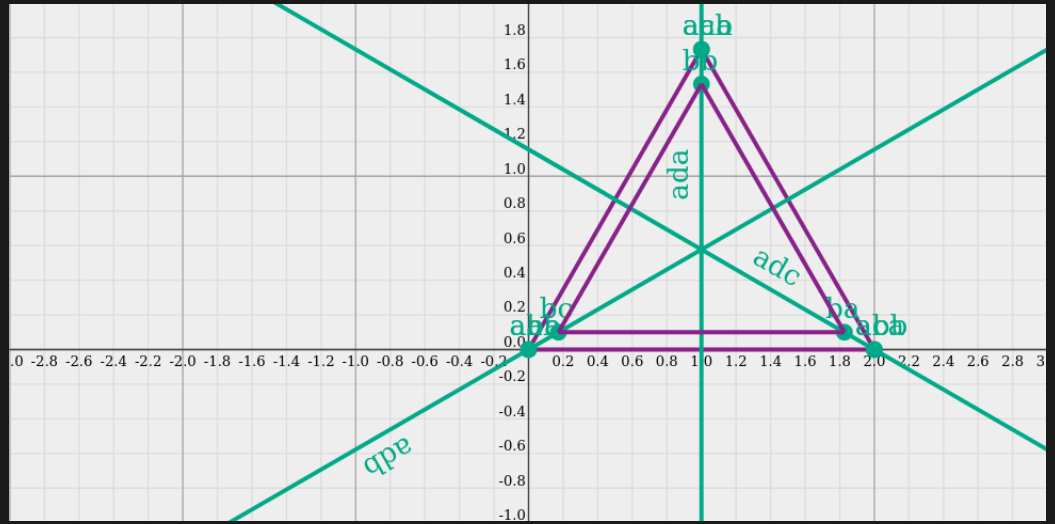
- Exist at the point that rays meet inside of a contour.
- Always emit a ray away from the input rays, along the bisector of the input rays(in the concave case).

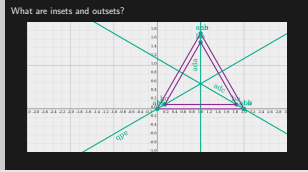
Slicing Primitives: Interior Nodes

- Exist at the point that rays meet inside of a contour.
- Always emit a ray away from the input rays, along the bisector of the input rays(in the concave case).

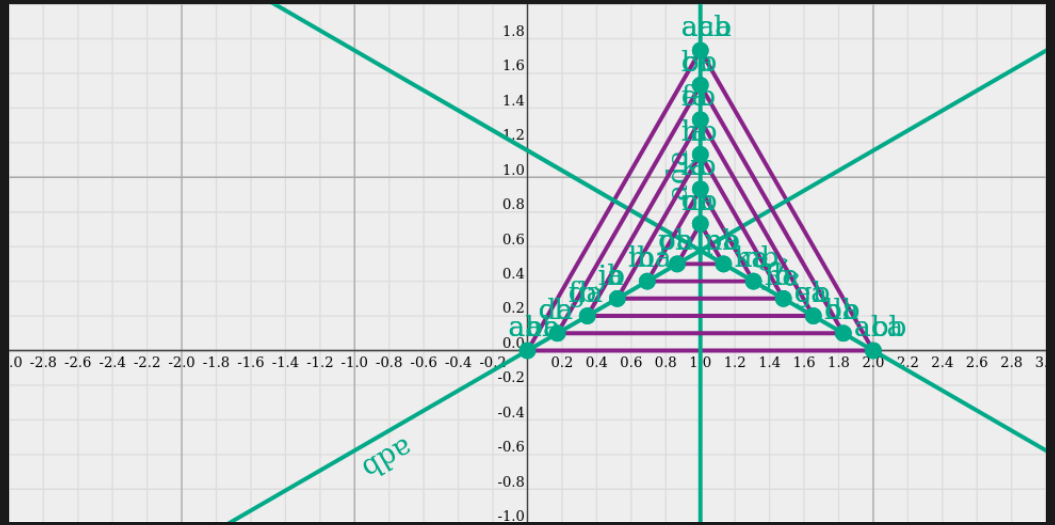
interior nodes are the joints in our straight skeleton

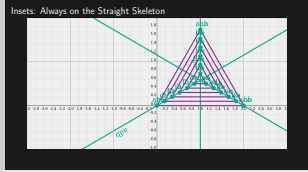
What are insets and outsets?





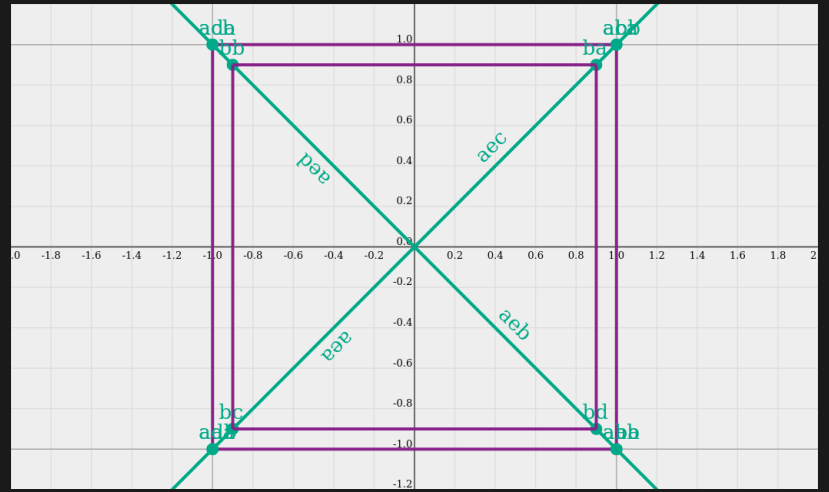
Insets: Always on the Straight Skeleton

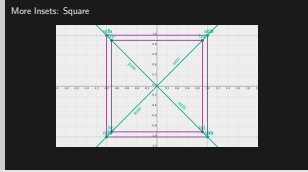




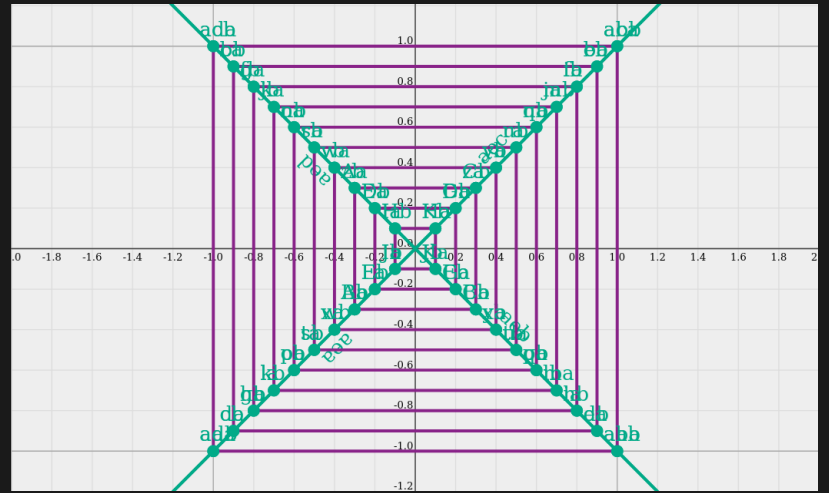
straight skeleton: the line segments on which we start and stop our inset line segments we are drawing the inset an object

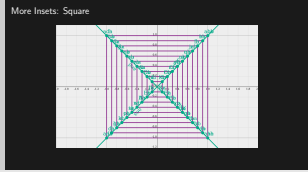
More Insets: Square





More Insets: Square



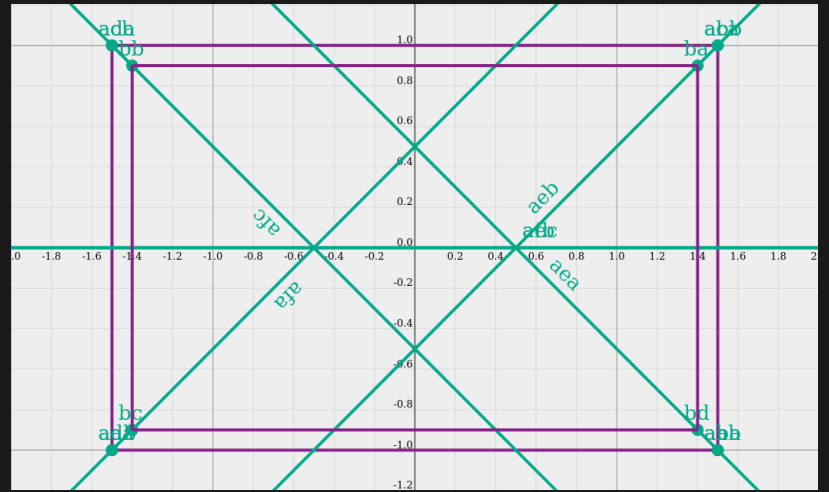


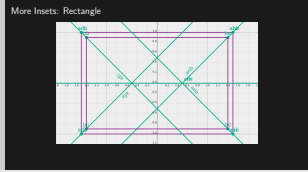
Why is any of this complicated?

Straight Skeletons

Why is any of this complicated?

More Insets: Rectangle





hslice's geometry engine is written to generate straight skeletons, for the purpose of doing insets and outsets.

Drawing the Straight Skeleton of a Convex Contour

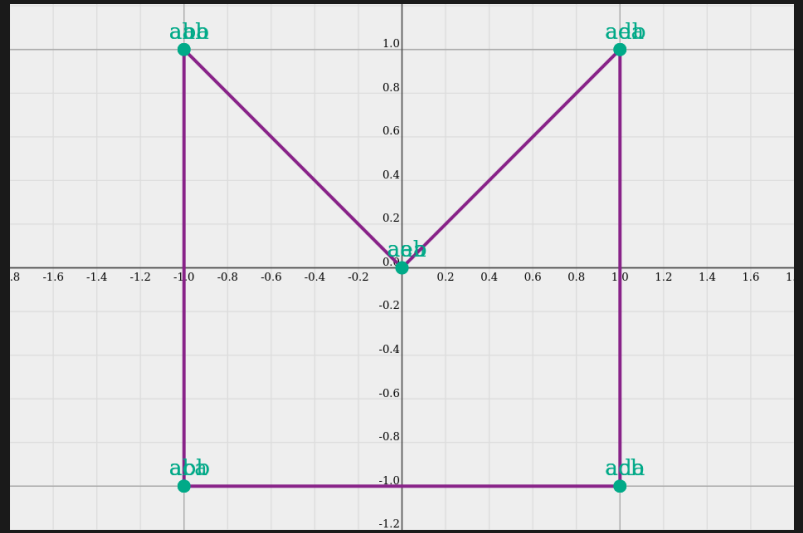
- Examine the collision of all of the rays emitted from nodes in the object.
- Determine the collision that has the shortest distance from its input nodes.
- Remove the input nodes from your list to be searched, and replace it with an interior node.
  - For two input nodes output of this new interior node will be anticollinear with the bisector of the input nodes.
- Wash, Rinse, Repeat.

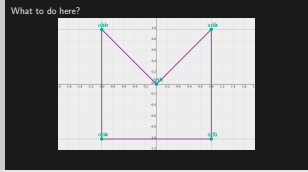
Drawing the Straight Skeleton of a Convex Contour

- Examine the collision of all of the rays emitted from nodes in the object.
- Determine the collision that has the shortest distance from its input nodes.
- Remove the input nodes from your list to be searched, and replace it with an interior node.
  - For two input nodes output of this new interior node will be anticollinear with the bisector of the input nodes.
- Wash, Rinse, Repeat.

repeat: our definition of convex: If you put a baloon around it, the baloon would touch all of the points of the contour.go back to the last slide, and solve it in-front of everyone

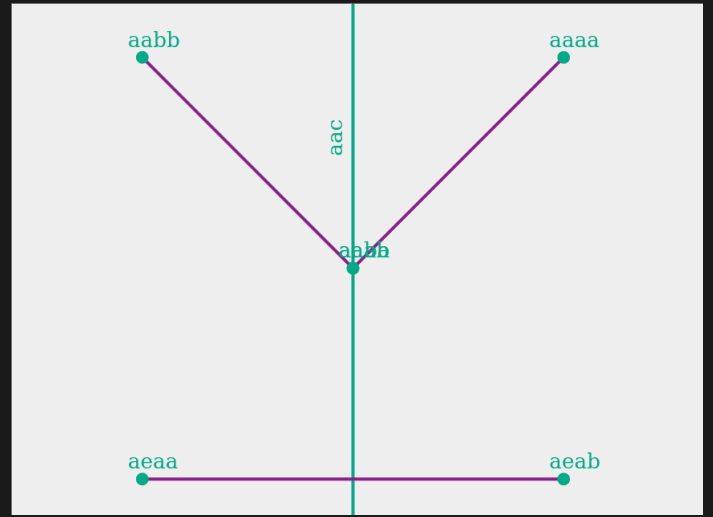
What to do here?

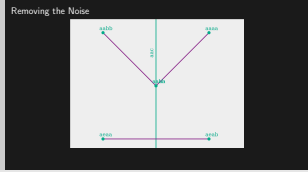




Explain basically what a motorcycle is here. repeat our definition of convex.

Removing the Noise





Explain what we have removed, and what remains

Slicing Primitives: Motorcycles

- Emitted from at the point two line segments intersect forming a convex angle from the perspective of the inside of the contour.
- Can collide with an exterior line, an exterior node, or another motorcycle.
- For a motorcycle that hits the other side un-obstructed, the point the straight skeleton passes through the motorcycle, and the lines the skeleton must enter/exit are calculable.
- Interesting Attributes:
  - Has a speed (speeding motorcycles!).

Slicing Primitives: Motorcycles

- Emitted from at the point two line segments intersect forming a convex angle from the perspective of the inside of the contour.
- Can collide with an exterior line, an exterior node, or another motorcycle.
- For a motorcycle that hits the other side un-obstructed, the point the straight skeleton passes through the motorcycle, and the lines the skeleton must enter/exit are calculable.
- Interesting Attributes:
  - Has a speed (speeding motorcycles!).

Speeding motorcycles are impossible to search for on google.

Speeds

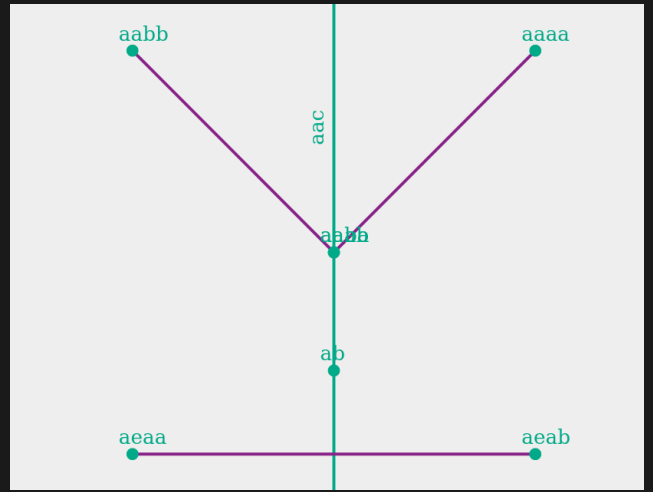
- Motorcycle Speed: The distance along the motorcycle between the point the motorcycle is emitted, and where it would intersect with a line that is parallel to, and one unit away from one of the motorcycle's input lines segments.
- Wall Collision Speed: The distance along the motorcycle between the point the motorcycle hits the wall, and where it would intersect with a line that is parallel to, and one unit away from the wall.

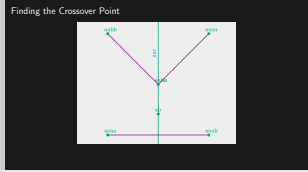
Speeds

- Motorcycle Speed: The distance along the motorcycle between the point the motorcycle is emitted, and where it would intersect with a line that is parallel to, and one unit away from one of the motorcycle's input lines segments.
- Wall Collision Speed: The distance along the motorcycle between the point the motorcycle hits the wall, and where it would intersect with a line that is parallel to, and one unit away from the wall.

So, in the case of being emitted from a 90 degree angle, giving us a 45 degree angle between where the motorcycle leaves a line segment, and where it hits a parallel line segment. that lets us use asquared + bsquared = csquared, which gives us a speed of 1.41.

Finding the Crossover Point





sqrt (2) == 1.41, multiply the two points, then add.

Finding the Crossover Point

The crossover point is the point which the straight skeleton must pass through between two cells.

To find the crossover point:

- Determine the speed of the emitted motorcycle.
- Determine the speed of the object it collided with.
- Place a point along the motorcycle where the ratio of the distance between the base of the motorcycle and the point of impact has the same ratio as the speeds do. Example:
  - For the last shown object, the motorcycle is emitted from a 90 degree angle.
  - A line parallel to one of it's input line segments and 1 unit away would cross the motorcycle sqrt(2), or 1.41 units away, giving it a speed of 1.41.
  - A the wall intersected is at a 90 degree angle to the motorcycle, so would have a speed of 1.
  - So the distance between the motorcycle base and the point is 1.41 times the distance between the point and where the motorcycle intersects the target line.

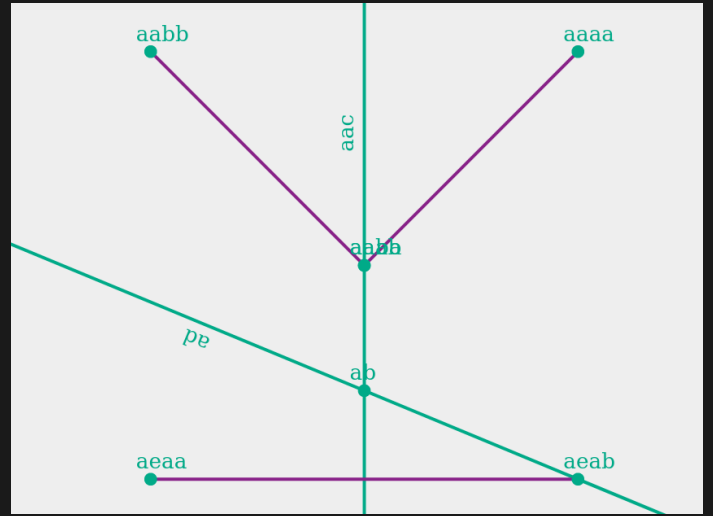
Finding the Crossover Point

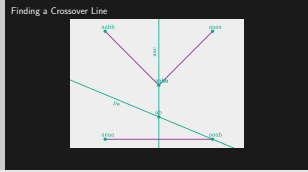
The crossover point is the point which the straight skeleton must pass through between two cells.

To find the crossover point:

- Determine the speed of the emitted motorcycle.
- Determine the speed of the object it collided with.
- Place a point along the motorcycle where the ratio of the distance between the base of the motorcycle and the point of impact has the same ratio as the speeds do. Example:
  - For the last shown object, the motorcycle is emitted from a 90 degree angle.
  - A line parallel to one of it's input line segments and 1 unit away would cross the motorcycle sqrt(2), or 1.41 units away, giving it a speed of 1.41.
  - A the wall intersected is at a 90 degree angle to the motorcycle, so would have a speed of 1.
  - So the distance between the motorcycle base and the point is 1.41 times the distance between the point and where the motorcycle intersects the target line.

Finding a Crossover Line





sqrt (2) == 1.41, multiply the two lines, then add.

Finding the Crossover Line

The crossover lines are the lines which the straight skeleton must pass through on both sides of our crossover point.
To find the crossover line:

- Determine the speed of the emitted motorcycle.
- Determine the speed of the object it collided with.
- Place a line where the ratio of the distance between the line segment on that side of the motorcycle and the line it impacts has the same ratio as the speeds do. Example:
  - For the last shown object, the motorcycle is emitted from a 90 degree angle.
  - A line parallel to one of it's input line segments and 1 unit away would cross the motorcycle sqrt(2), or 1.41 units away, giving it a speed of 1.41.
  - A the wall intersected is at a 90 degree angle to the motorcycle, so would have a speed of 1.
  - So the distance between the motorcycle base and the new line is 1.41 times the distance between the new line and where the motorcycle intersects the target line.

Finding the Crossover Line

The crossover lines are the lines which the straight skeleton must pass through on both sides of our crossover point.
To find the crossover line:

- Determine the speed of the emitted motorcycle.
- Determine the speed of the object it collided with.
- Place a line where the ratio of the distance between the line segment on that side of the motorcycle and the line it impacts has the same ratio as the speeds do. Example:
  - For the last shown object, the motorcycle is emitted from a 90 degree angle.
  - A line parallel to one of it's input line segments and 1 unit away would cross the motorcycle sqrt(2), or 1.41 units away, giving it a speed of 1.41.
  - A the wall intersected is at a 90 degree angle to the motorcycle, so would have a speed of 1.
  - So the distance between the motorcycle base and the new line is 1.41 times the distance between the new line and where the motorcycle intersects the target line.

Slicing Primitives: Cells

- Have a non-zero number of sides, composed of multiple connected line segments.
- Are separated from other cells by motorcycles.

Slicing Primitives: Cells

- Have a non-zero number of sides, composed of multiple connected line segments.
- Are separated from other cells by motorcycles.

Implications?

- Because the crossover point and lines can be calculated directly from just the input segments of the motorcycle, and the object collided with:
  - We can calculate the straight skeletons of the two sides separately, giving us multiple cells. Parallelism, and reduced work on both sides!
  - When calculating using cells, error in calculation is not propogated across the divide.
- Because PGA has operations for both finding a point at a ratio between two points, and for finding a line at a ratio between two other lines, The crossover finding functions are easy to implement.
- Crossover calculations do not work if one of the interior nodes of the skeleton of one side crosses over to the other side. for this situation, see Christopher Tscherne's paper referenced earlier.

Implications?

- Because the crossover point and lines can be calculated directly from just the input segments of the motorcycle, and the object collided with:
  - We can calculate the straight skeletons of the two sides separately, giving us multiple cells. Parallelism, and reduced work on both sides!
  - When calculating using cells, error in calculation is not propogated across the divide.
- Because PGA has operations for both finding a point at a ratio between two points, and for finding a line at a ratio between two other lines, The crossover finding functions are easy to implement.
- Crossover calculations do not work if one of the interior nodes of the skeleton of one side crosses over to the other side. for this situation, see Christopher Tscherne's paper referenced earlier.

How does PGA help?

Projective Geometric Algebra
How does PGA help?

PGA Primitives: 2D Projective Point

- Based on values in three basis vectors pairs:
  - (e0, e1)
  - (e0, e2)
  - (e1, e2)
- Interesting Operations:
  - Add two points: get a point in-between them
  - Multiply both points by values a,b, then add them: get a point a/b between the two points
  - Multiply two points: get a line between them, and the distance between the points

PGA Primitives: 2D Projective Point

- Based on values in three basis vectors pairs:
  - (e0, e1)
  - (e0, e2)
  - (e1, e2)
- Interesting Operations:
  - Add two points: get a point in-between them
  - Multiply both points by values a,b, then add them: get a point a/b between the two points
  - Multiply two points: get a line between them, and the distance between the points

PGA Primitives: 2D Projective Line

- Based on values in three basis vectors:
  - e0 – TranslationISH
  - e1 – xInterceptISH
  - e2 – yInterceptISH
- Interesting Operations:
  - Add two lines: get the line in-between them
  - Multiply both lines by values a,b, then add them: get a line a/b between the two lines
  - No e1? Parallel to the X axis.
  - Ro e2? Parallel to the Y axis.
  - Use the translationISH and an interceptISH to get a real intercept.
- Interesting Properties:
  - Has a direction.

PGA Primitives: 2D Projective Line

- Based on values in three basis vectors:
  - e0 – TranslationISH
  - e1 – xInterceptISH
  - e2 – yInterceptISH
- Interesting Operations:
  - Add two lines: get the line in-between them
  - Multiply both lines by values a,b, then add them: get a line a/b between the two lines
  - No e1? Parallel to the X axis.
  - Ro e2? Parallel to the Y axis.
  - Use the translationISH and an interceptISH to get a real intercept.
- Interesting Properties:
  - Has a direction.

What about floating point error?

Projective Geometric Algebra

What about floating point error?

Floating Point and Projective Geometric Algebra

- We implemented PGA using IEEE754 double precision.
  - One sign bit.
  - 11 exponent bits.
  - 52 fraction bits.
- X86 has three multiply modes:
  - Give me the closest answer above the 'real' answer.
  - Give me the closest answer to the 'real' answer.
  - Give me the closest answer below the 'real' answer.
- Unit of Least Precision:
  - The least significant bit of a value.

Floating Point and Projective Geometric Algebra
- We implemented PGA using IEEE754 double precision.
  - One sign bit.
  - 11 exponent bits.
  - 52 fraction bits.
- X86 has three multiply modes:
  - Give me the closest answer above the 'real' answer.
  - Give me the closest answer to the 'real' answer.
  - Give me the closest answer below the 'real' answer.
- Unit of Least Precision:
  - The least significant bit of a value.

Define a ULP here

Acquiring Error Bars

- Every time you perform a calculation:
  - Get the closest answer to the 'real' answer.
  - Get the closest answer above the 'real' answer.
  - Store the closest answer to the 'real' answer.
  - Store the unit of least precision of the closest answer above the 'real' answer.

Acquiring Error Bars

- Every time you perform a calculation:
  - Get the closest answer to the 'real' answer.
  - Get the closest answer above the 'real' answer.
  - Store the closest answer to the 'real' answer.
  - Store the unit of least precision of the closest answer above the 'real' answer.

Utilizing Error Bars

- Every time you perform a geometric operation, take into account these errors:
  - Comparing where a line intersects with the end of a line segment?
    - Use the ULP(Unit of Least Precision)s to decide on the size of a 'hit circle' around the endpoint
    - Check distance of the found intersection from that point.
    - Intersections between two line segments at the same point will probably have different sized hit circles!
  - Checking whether two lines are collinear?
    - Use the ULPs to decide on the size of deviation that's possible.
  - EACH situation may have different ways to interpret!
  - When in doubt, add a fudge multiplier, and think through your operations!

Utilizing Error Bars

- Every time you perform a geometric operation, take into account these errors:
  - Comparing where a line intersects with the end of a line segment?
    - Use the ULP(Unit of Least Precision)s to decide on the size of a 'hit circle' around the endpoint
    - Check distance of the found intersection from that point.
    - Intersections between two line segments at the same point will probably have different sized hit circles!
  - Checking whether two lines are collinear?
    - Use the ULPs to decide on the size of deviation that's possible.
  - EACH situation may have different ways to interpret!
  - When in doubt, add a fudge multiplier, and think through your operations!

Total Cheating

- Abusing Laziness:
  - Haskell is a lazy language.
  - ULPs allow you to find error bars, including error amounts that might be undesireable.
  - Solution? If the error value is too large, do the calculation in a fixed point library!

Total Cheating

- Abusing Laziness:
  - Haskell is a lazy language.
  - ULPs allow you to find error bars, including error amounts that might be undesireable.
  - Solution? If the error value is too large, do the calculation in a fixed point library!

How do I know any of that worked?

Property Tests

How do I know any of that worked?

Property Tests

Property tests test for a “property” of calling a function with random data.
Examples:

- The sum of the three angles of any triangle is 180.
- All squares have four sides.
- All regular polygons have no motorcycles.

Property Tests

Property tests test for a “property” of calling a function with random data.
Examples:
- The sum of the three angles of any triangle is 180.
- All squares have four sides.
- All regular polygons have no motorcycles.

Property Testing Geometry

To property test geometry, you must do a few things:

- Create a function from a set of (radian, distance) and a center point to a contour.
- Call that function from your property test, and test properties!
- Think:
  - The point of the interior node of a triangle is always inside the triangle..
  - A square always has one interior node.
  - A rectangle always has two interior nodes.
  - An inset unit polygon retains the number of exterior nodes as it shrinks.

Idea Stolen From:

https://stackoverflow.com/questions/8997099/algorithm-to-generate-random-2d-polygon

Property Testing Geometry

To property test geometry, you must do a few things:
- Create a function from a set of (radian, distance) and a center point to a contour.
- Call that function from your property test, and test properties!
- Think:
  - The point of the interior node of a triangle is always inside the triangle..
  - A square always has one interior node.
  - A rectangle always has two interior nodes.
  - An inset unit polygon retains the number of exterior nodes as it shrinks.

Idea Stolen From:
https://stackoverflow.com/questions/8997099/algorithm-to-generate-random-2d-polygon

Review

To summarize:

- In order for 3d printing to help more people, we need to focus on slicers.
- More parallel algorithms for slicing are possible.
- Projective Geometric Algebra can be used with less precision loss than linear algebra.
- Property testing is necessary, to keep all of this math working as you develop it.
- I want you to write a slicer in less than the 3 years this has taken me.

Review

To summarize:
- In order for 3d printing to help more people, we need to focus on slicers.
- More parallel algorithms for slicing are possible.
- Projective Geometric Algebra can be used with less precision loss than linear algebra.
- Property testing is necessary, to keep all of this math working as you develop it.
- I want you to write a slicer in less than the 3 years this has taken me.

Questions?

email: julia.longtin@gmail.com

-Fin-
Questions?
email: julia.longtin@gmail.com

I don't care if my slicer is the one we get to the destination with, i'm just trying to get us closer.