Contents

List of Figures

About

This document contains notes about the *Giddy Machinist* project.

1 Straight Skeleton Generation

The straight skeleton is generated via a two-phase approach. At first, the so-called motorcycle graph is generated, then the actual straight skeleton. One algorithm is described in [1].

There are single-phase and two-phase approaches. The two phase approach seems more robust for handling degenerate geometry, for example symmetric polygons, which are very common in practical usage. The two phase approach begins with the motorcycle tracing, which creates clear boundaries for non-convex polygons.

Another advantage to the two phase approach is that principally, the time complexity of the 2nd phase is greatly reduced, as all events that need to be tracked are of time complexity $O(N + M)$, as there are only collapse events for each edge, as well as the wavefronts opposite of a motorcycle, that need to be tracked.

1.1 The Motorcycle Graph

For a simple polygon, motorcycles are only emitted by reflex[1] vertices. The motorcycles are angular bisector rays, and may crash into each other or the polygon edges as in the game *Tron*. A motorcycle trace is terminated when it reaches an edge or other motorcycle trace. A degenerate case, which must be handled for real-world geometry, is when two or more motorcycles collide at the same point, in which case a new motorcycle must be emitted.

1.1.1 Events

Three different events may happen during motorcycle tracing:

Edge Crash The edge crash is the simplest event, and happens when a motorcycle hits a polygon edge. The edge is split (unless the hit is at an existing vertex), and a new vertex is inserted.

Trace Crash When a motorcycle hits the trace of another motorcycle, it is terminated, and a vertex is generated.

Merge Crash When two motorcycles reach the same point in space at the same time, they merge, and may emit one or two new motorcycles in new directions, depending on the crash geometry. This kind of crash can be ignored, and a trace crash can be used instead.

1.1.2 Special Considerations

The times for potential crashes do not change during tracing, but new motorcycles may be generated during merges.

[1] what exactly is a reflex vertex?

Events are treated as simultaneous if they occur within some limit of location and time. As motorcycle events are point-like in nature, there is no danger of invalidating the graph.

Collinear Motorcycles A special degenerate case can occur when two motorcycles are collinear, but in opposite directions. In such a case, one of the motorcycles may be discarded. This commonly occurs in CAD data with aligned holes. Cases can occur when only one endpoint is shared, but the other is a trace crash site.

A special difficulty occurs when a motorcycle terminates near an outline vertex, and consequently slightly changes direction as it's nudged into the vertex.

1.1.3 Tracing Algorithm

The implementation is outlined in Algorithm 1. Large performance improvements could be made by eliminating redundant intersection queries.

1.2 Wavefront Propagation

After the motorcycle graph is generated, the wavefront propagation can be started. Each edge emits a wavefront, which can either collapse, or be split by a motorcycle.

1.2.1 Wavefront Events

The following events may occur during wavefront propagation:

Wavefront Collapse The collapse is the simplest event. A wavefront collapses into a vertex, which emits a new spoke bisecting the neighbouring wavefronts.

Motorcycle Split The split occurs along a motorcycle trace, when the motorcycle would hit the opposing face. As the wavefronts are already pre-split along the motorcycle trace, only minor topological changes need to happen.

Motorcycle Branch A branch occurs when a motorcycle trace reaches a point where another motorcycle will crash into it. The wavefront is split along the incoming motorcycle trace.

Motorcycle Merge A merge corresponds to the motorcycle merge crash. This event splits the combined wavefront into two sub-wavefronts.

Reverse Merge Occurs when a merge vertex is reached by the opposing spoke before it is reached by the motorcycles that created it. New wavefronts are inserted between the incoming motorcycles.

Algorithm 1 [OBSOLETE] Computation of motorcycle graph.

```
Require: simple input polygon
  for each vertex do
    create motorcycle structure
  end for
  repeat
    for each active motorcycle do
      record crash times with edges
      record crash times with active motorcycles
      record crash times with terminated motorcycles
    end for
    sort crash events by time
    pick first crash
    assemble array off all crash events within merge threshold of location and time from first
  crash
    determine crash subtypes: wall, trace, merge
    if contains wall crash then
      all traces terminate at crash site
      if not at vertex then
        split edge, insert new vertex
      end if
    else if contains trace crash then
      create new vertex, terminate crashed motorcycles
    else
      create new vertex, terminate involved motorcycles
      if non-convex sub-polygon remains then
        start new 'escaped' motorcycle
      end if
    end if
  until no more active motorcycles
```

Algorithm 2 [OBSOLETE] Wavefront Initialization.

Require: simple input polygon
Require: motorcycle graph
  for each motorcycle **do**
    create motorcycle spoke and anti-spoke
  end for
  for each vertex on outline **do**
    create spoke if vertex is convex, and no motorcycle spoke or anti-spoke exists
  end for
  for each each spoke starting at $t = 0$ **do**
    create wavefront
    attach wavefront
  end for
  generate emission events
  for each wavefront **do**
    compute collapse time
    add collapse event to event queue
  end for
  for each motorcycle spoke **do**
    if motorcycle starts on outline vertex **then**
      compute split event
    end if
    if motorcycle ends on outline vertex **then**
      for each other motorcycle crashed into trace **do**
        compute reverse branch event
      end for
      if motorcycle starts on merge vertex **then**
        compute reverse merge event
      end if
    end if
    if motorcycles ends and starts on outline **then**
      add earliest event to event queue
    else
      add split or reverse event to event queue, whichever exists
    end if
  end for

Algorithm 3 Collapse Event Handling

```
Add spokes belonging to wavefront, and neighbours, to changed spokes.
Remove wavefront's spokes from active list.
Terminate wavefront.
if both left and right spoke were already terminated then
    NOP
else
    if left or right spoke terminated then
        Use vertex for event.
    else
        Create new vertex at event location.
    end if
    Assign event vertex to unterminated spoke(s).
    if left and/or right spoke are motorcycle spokes then
        As spoke is being terminated, remove it from opposing wavefront (split) tracking.
    end if
    if neighbours aren't weakly convex OR both neighbours are the same object then
        ¿ terminate neighbour wavefronts or not ?
    else
        Create new spoke between neighbours.
        re-assign opposing spokes from wavefront to neighbours
    end if
end if
```

Algorithm 4 Split Event Handling

```
Add spokes to changed spokes.
Create split vertex.
As splitting spoke is being terminated, remove it from opposing wavefront (split) tracking.
create newLeftFront and newRightFront.
Terminate split wavefront.
if newLeftFront/newRightFront vestigial then
    Discard vestigial front(s) again.
end if
for each opposing spoke of split front do
    Re-assign tracking to newLeftFront/newRightFront accordingly.
end for
```

Reverse Branch Occurs when a branch vertex is reached by the opposing spoke before it is reached by the motorcycles that created it. New wavefronts are inserted between the incoming motorcycles.

Emit Contour When the offset/time of a contour is reached, the wavefront graph can easily be traversed to emit a polygon at this point.

The motorcycle branch and contour emission events are static, thus occur at fixed times that can be computed before the wavefront propagation is started. The collapse and split events, however, are dynamic, and need to be recomputed when certain changes in topology occur.

The reverse merge and branch events can also only be computed dynamically, as when the vertices are hit by backpropagating anti-spokes may depend on previous events.

1.2.2 Simultaneous Events

The complex topology of the wavefront graph results in many theoretically degenerate cases, which nevertheless occur in practical applications.

Unlike in the motorcycle tracing phase, degenerate events are not point-like. Resolving degenerate events must not invalidate the graph.

Fortunately, after the separate motorcycle phase, only two classes of degenerate events may occur:

Arc Collapse The arc collapse happens when multiple wavefronts collapse into a single point.

Opposing Wavefronts This case occurs when the neighbouring wavefronts to the one collapsing are anti-parallel. Solved by fast spokes which have pseudo-infinite speed.

Simultaneous collapses should be resolvable sequentially, as long as the temporary resulting wavefronts are allowed to collapse immediately.

Sequential resolution of events leads to some complicated logic, though, as the order of events may vary.

1.2.3 Opposing Wavefronts Collapse and Motorcycle Traces

When a motorcycle anti-spoke is involved in a degenerate collapse, care must be taken to propagate the anti-spoke correctly.

It appears that the anti-spoke can be dropped in such cases, as cycle-spoke is entering a convex sub-region. This can be detected at the cycle spoke if one of the attached wavefronts collapses, and the new angle bisector 'cuts off' the motorcycle trace.

The test for a collapsing anti-spoke is as follows: if the new wavefront travels in the same direction as the anti-spoke, keep it, and split the new wavefront accordingly, otherwise terminate anti-spoke.

It seems, at this point, that degenerate cases can be handled gracefully, and locally, simply by removing the offending wavefronts from the active list.

2 FDM Features

2.1 Perimeters and Infill

Typically, it is desired to trace the perimeters of a layer, with one or more traces, and fill the inner volume with a regular pattern, such as alternating straight lines, or a honeycomb.

The outer trace is inset half the extrusion width from the outline, and every further trace a full extrusion width. After the innermost perimeter trace, the patterned area is also inset a full line width.

2.1.1 Thin Walls

A thin wall is a section of geometry which is too narrow for a deposited loop of material. It can occur as an original feature of the outline, or as the result of polygon insetting, and the question is how to identify these areas.

It may be possible to look at the history of the wavefront tracing, to peek into the future at any given inset, and determine if a given polygon segment will border a thin wall or not.

By keeping track of the wavefront associated with an emitted line segment, it can be determined if the wavefront terminates before the full trace width is reached. In such a case, the area may be infilled with a pattern, or the skeleton can be traced with a single extrusion. The extrusion width may even be varied, according to the timing of the straight skeleton.

2.1.2 Overfill/Underfill in Perimeter Extrusion

Thin perimeters are a variant of thin walls. For a perimeter extrusion, two possible scenarios exist:

Underfill The trace is not even generated, but the previous perimeter doesn't fully fill the area.

Overfill The trace is generated, but would result in overfilling.

Both may be detected by tracking half an extrusion width increments.

2.2 Sparse Infill, Top and Bottom Layers

It may be desired to not fully fill the volume, in which case a sparse infill is used, which has traces wider apart than the extrusion width.

However, it is usually desired to fully fill some number of the top and bottom layers. To find out which areas of the layer are top and bottom, respectively, the infill area is compared to the top and bottom layers with a boolean intersection.

2.3 Deposition Model

Printing occurs at a feed rate v.

2.3.1 Nozzle Parameters

The nozzle is assumed to have a certain diameter d_N. From the nozzle, the filament extrudes with a free extrusion diameter d_F. The free extrusion diameter depends on other parameters, such as extrusion speed. It is slightly bigger than d_N. A default of

$$d_F = d_N + 0.08 \text{ mm}$$

seems to work ok[2].

The behavior of d_F should be investigated further, do determine how temperature, extrusion rate, etc., affect it.

From d_F the free extrusion area

$$A_F = \frac{1}{4}\pi d_F$$

can be computed.

The extruder flow rate $Q_E = A_E \cdot v_E$, where v_E is the extruder speed, and A_E is the filament cross sectional area, given by $A_E = \frac{1}{4}\pi d_E$. As flow is constant, $Q_E = Q_F$, therefore $Q_F = Q_E$, and

$$v_F = A_E/A_F \cdot v_E.$$

2.3.2 Layer height

Layer height h_L can be between $20\% - 80\%$ of nozzle diameter[3].

[2] Based on Josef Prusa's reprap calculator values.

[3] by anecdotal reference

2.3.3 Layer Width

The nominal layer width w_L should be at least twice h_L. The suggested approach is that the flow rate should be matched to the feed rate, as *slic3r* does. This means that $w_L \cdot h_L = A_F$, therefore

$$w_L = \frac{A_F}{h_L}.$$

Layer width should not be excessively large, or it may overflow the nozzle left and right.

Using this formula for a 0.35 mm nozzle with 0.2 mm layer height generates a wide layer of 0.72 mm, instead of the more usual 0.5 – 0.6 mm computed by other slicers.

2.4 Overhangs

What affects curling on overhangs? arc radius? speed? layer height? cooling? surface tension?

Maybe lowering the print nozzle during overhang prints can counter the curling.

2.5 Overhang Support

2.5.1 Gantry Support

As in [2].

2.5.2 Lamella Support

2.6 Start/End Overlap on Perimeters

2.7 Infill generation

2.8 Thermal Properties of PLA

Table 1 contains the specific heat values for PLA. A glass transition temperature of 57 °C is indicated. This corresponds with a king in the specific heat graph.

Based on flow rate and specific heat capacity, the amount of heat energy required to melt the filament can be computed, and it can be used for better heat regulation.

3 Fixed Point Arithmetic Polygon Offsetting

Fixed point math could get rid of numeric issues that plague floating point implementations for polygon offsetting.

Table 1: Specific heat of PLA, from [3].

Temperature °C	Specific Heat (C_p) $\frac{\text{J}}{\text{kg} \cdot \text{K}}$
52	1483
55	1590
58	1725
60	1804
62	1851
65	1880
70	1901
100	1955
140	1994
160	2020
190	2060
230	2114

Based on the millimeter as a base unit, a 16.16 fixed point representation results in a work area of 65 by 65 meters, with a resolution of about 15 nanometers, which is quite sufficient for the intended target of FDM printing and small scale milling.

Furthermore, arbitrary scaling can be applied to increase the working area, or to increase resolution.

3.1 Multi-Precision Integer Arithmetic

The use of multi-precision integer arithmetic allows good control of rounding artifact, and some operations are even exact. Multiplication and addition can be performed without penalty, only division and square root, of the common operations in this context, are rounded.

Even though the final result of some computation is almost certainly rounded, the rounding operations can be factored out to be performed last, thus preventing the accumulation of rounding errors.

3.2 3D Triangle-Plane Intersection

The Input for slicing is an STL file with floating point vertices. Several floating point vertices may map to a single fixed-point vertex. The line segment sliced by a plane is determined via the edges of the triangle intersecting the plane. A triangle intersects a plane in two points, which come from two different edges.

For given vertices A and B, the intersection point of the plane and the edges AB and BA must be the same point.

3.3 2D Line-Line Intersection

The intersection of two lines in 2D must also be exact. Reversing any of the lines must yield the same intersection point.

For given points P, Q and segments R, S:

$$P + u \cdot R = Q + v \cdot S = X$$

The times are given by[4]

$$\begin{aligned} d &= R \otimes S \\ u &= \frac{(Q - P) \otimes S}{d} \\ v &= \frac{(P - Q) \otimes R}{d} \end{aligned}$$

Alternatively, as described [4], via the equation system's determinant:

$$X = \frac{R(Q \otimes S) - S(P \otimes R)}{R \otimes S}.$$

3.4 2D Line Point Distance

The perpendicular distance vector D of line segment AB to a point P:

$$\begin{aligned} D &= (P - A) - AB\frac{(P - A) \cdot (AB)}{AB \cdot AB} \\ D &= \frac{AP(AB \cdot AB) - AB(AP \cdot AB)}{AB \cdot AB} \\ x_D &= \frac{x_{AP}y_{AB}^2 - y_{AP}x_{AB}y_{AB}}{x_{AB}^2 + y_{AB}^2} \\ y_D &= \frac{y_{AP}x_{AB}^2 - x_{AP}x_{AB}y_{AB}}{x_{AB}^2 + y_{AB}^2}. \end{aligned}$$

[4] $A \otimes B$ in this context is a pseudo cross product, equivalent to $x_A y_B - x_B y_A$. The extended cross product identities do not apply, but it is anticommutative, and distributive over addition and scalar multiplication.

$$
\begin{aligned}
D &= (P-A) - AB\frac{(P-A)\cdot(AB)}{AB\cdot AB} \\
D &= \frac{AP(AB\cdot AB) - AB(AP\cdot AB)}{AB\cdot AB} \\
x_D &= \frac{x_{AP}y_{AB}^2 - y_{AP}x_{AB}y_{AB}}{x_{AB}^2 + y_{AB}^2} \\
y_D &= \frac{y_{AP}x_{AB}^2 - x_{AP}x_{AB}y_{AB}}{x_{AB}^2 + y_{AB}^2}.
\end{aligned}
$$

If the order of multiple points by distance has to be determined, this can be done exactly with the numerator of the above fraction. For different edges AB and UV, where D' is the distance fraction's numerator, the distance comparison becomes

$$D'_{AB}(UV\cdot UV) \quad <=> \quad D'_{UV}(AB\cdot AB).$$

The same rearrangement also works for different points, thus, order can be determined exactly in a fixed point context with multi-precision integers, even for multiple lines and multiple points.

3.5 Vector Scaling

As the intersection test yields parameters as a fraction of 64-bit integers, it must be examined if storing time in that format is reasonable.

For two points A, B, the equations $A+t\cdot AB$ and $B+(1-t)\cdot BA$ must equal. This is easily accomplished if time is an integer fraction. The question is, how accurate does time have to be. Is a 32-bit fraction sufficient?

To answer the question, at first, let's look at a line $R = (2^{31}-1, 0)$. With a 32-bit time value between 0..1, this vector can be scaled to ANY point on the line, and at the same time the full range of possible values.

Next, let's look at $S = (1, 0)$. With a 32-bit value between $0..2^{31}-1$, this vector can be scaled to the full range of allowable values.

Thus, it is necessary and sufficient to be able to represent $[2^{-31}, 2^{31})$ to scale any line to any point on itself. This can be accomplished with a 32-bit fraction. A 64-bit fraction is not needed, as the low bits would be rounded off anyway, and can be safely ignored.

No velocity vector can be faster than 2^{31}, either.

3.6 2D Angular Bisectors

The angular bisectors must also exactly represent the intersection point of the two offset source edges, for any given time. For given source vertices A, B, C, the angular bisector S bisects

edges E_{AB}, E_{BC}:

$$S = E_{BC}\frac{N_{AB}\cdot N_{AB}}{N_{AB}\cdot E_{BC}} + E_{AB}\frac{N_{BC}\cdot N_{BC}}{N_{BC}\cdot E_{AB}}.$$

For a given t, $X = S\cdot t$ must give the exact position of X.

To be exact, during wavefront propagation, two neighbouring bisectors must reach their intersection point at the same time.

The normal N is composed of $N = (-y_E, x_E)/||E||$, so[5]

$$\begin{aligned}
S &= \frac{E_{BC}}{||E_{AB}||}\frac{E_{AB}\cdot E_{AB}}{N'_{AB}\cdot E_{BC}} + \frac{E_{AB}}{||E_{BC}||}\frac{E_{BC}\cdot E_{BC}}{N'_{BC}\cdot E_{AB}} \\
S &= \frac{E_{BC}}{||E_{AB}||}\frac{E_{AB}\cdot E_{AB}}{E_{AB}\otimes E_{BC}} + \frac{E_{AB}}{||E_{BC}||}\frac{E_{BC}\cdot E_{BC}}{E_{BC}\otimes E_{AB}} \\
S &= E_{BC}\frac{||E_{AB}||}{E_{AB}\otimes E_{BC}} + E_{AB}\frac{||E_{BC}||}{E_{BC}\otimes E_{AB}} \\
S &= E_{BC}\frac{||E_{AB}||}{E_{AB}\otimes E_{BC}} - E_{AB}\frac{||E_{BC}||}{E_{AB}\otimes E_{BC}} \\
S &= \frac{E_{BC}||E_{AB}|| - E_{AB}||E_{BC}||}{E_{AB}\otimes E_{BC}}
\end{aligned}$$

A numeric problem occurs if $E_{AB}\times E_{BC}$ is zero. In that case, the lines are either parallel or anti-parallel to within numerical limits. If they are parallel, the term $E_{BC}||E_{AB}|| - E_{AB}||E_{BC}||$ tends towards zero as well as the denominator, and a well-behaved bisector of the form

$$S = \frac{N_{AB} + N_{BC}}{2}$$

results. In the other case, the bisector is collinear with both vectors, and of infinite velocity. The anti-parallel case results in the direction $S' = E_{BC}||E_{AB}|| - E_{AB}||E_{BC}||$.

[5] Substituting inversely, with $E = ||E||\cdot(y_N, -x_E)$ yields a very simple simplification, but with numeric pitfalls

$$\begin{aligned}
S &= \frac{E_{BC}}{||E_{BC}||}\frac{N_{AB}\cdot N_{AB}}{N_{AB}\times N_{BC}} + \frac{E_{AB}}{||E_{AB}||}\frac{N_{BC}\cdot N_{BC}}{N_{BC}\otimes N_{AB}} \\
S &= N_{BC}\frac{N_{AB}\cdot N_{AB}}{N_{AB}\otimes N_{BC}} - N_{AB}\frac{N_{BC}\cdot N_{BC}}{N_{AB}\otimes N_{BC}} \\
S &= \frac{N_{BC} - N_{AB}}{N_{AB}\otimes N_{BC}}
\end{aligned}$$

3.6.1 Motorcycle-Motorcycle Crash

Carrying on to the intersection of bisectors ABC, UVW:

$$\begin{aligned}
R &= \frac{E_{BC}||E_{AB}|| - E_{AB}||E_{BC}||}{E_{AB} \otimes E_{BC}} \\
S &= \frac{E_{VW}||E_{UV}|| - E_{UV}||E_{VW}||}{E_{UV} \otimes E_{VW}} \\
R \otimes S &= \frac{E_{BC}||E_{AB}|| - E_{AB}||E_{BC}||}{E_{AB} \otimes E_{BC}} \otimes \frac{E_{VW}||E_{UV}|| - E_{UV}||E_{VW}||}{E_{UV} \otimes E_{VW}} \\
R \otimes S &= \frac{(E_{BC}||E_{AB}|| - E_{AB}||E_{BC}||) \otimes (E_{VW}||E_{UV}|| - E_{UV}||E_{VW}||)}{(E_{AB} \otimes E_{BC}) \cdot (E_{UV} \otimes E_{VW})} \\
X &= \frac{(E_{BC}||E_{AB}|| - E_{AB}||E_{BC}||) \cdot (V \otimes (E_{VW}||E_{UV}|| - E_{UV}||E_{VW}||))}{(E_{BC}||E_{AB}|| - E_{AB}||E_{BC}||) \otimes (E_{VW}||E_{UV}|| - E_{UV}||E_{VW}||)} \\
&- \frac{(E_{VW}||E_{UV}|| - E_{UV}||E_{VW}||) \cdot (B \otimes (E_{BC}||E_{AB}|| - E_{AB}||E_{BC}||))}{(E_{BC}||E_{AB}|| - E_{AB}||E_{BC}||) \otimes (E_{VW}||E_{UV}|| - E_{UV}||E_{VW}||)}
\end{aligned}$$

3.6.2 Motorcycle-Edge Crash

The crashing of bisector of ABC and edge UV:

$$\begin{aligned}
R &= \frac{E_{BC}||E_{AB}|| - E_{AB}||E_{BC}||}{E_{AB} \otimes E_{BC}} \\
S &= E_{UV} \\
R \otimes S &= \frac{E_{BC}||E_{AB}|| - E_{AB}||E_{BC}||}{E_{AB} \otimes E_{BC}} \otimes E_{UV} \\
X &= \frac{(E_{BC}||E_{AB}|| - E_{AB}||E_{BC}||) \cdot (V \otimes E_{UV}) - E_{UV}(B \otimes (E_{BC}||E_{AB}|| - E_{AB}||E_{BC}||))}{(E_{BC}||E_{AB}|| - E_{AB}||E_{BC}||) \otimes E_{UV}}
\end{aligned}$$

3.6.3 Motorcycle-Wavefront Split

The wavefront split is similar to the edge crash, except that the wavefront moves. The wavefront's velocity can be added to R.

$$
\begin{aligned}
\hat{N} &= \frac{(-y_E, x_E)}{||E||} \\
R' &= \frac{E_{BC}||E_{AB}|| - E_{AB}||E_{BC}||}{E_{AB} \otimes E_{BC}} \\
S &= R' - \hat{N}_{UV} \\
S &= E_{UV} \\
R \otimes S &= \frac{E_{BC}||E_{AB}|| - E_{AB}||E_{BC}||}{E_{AB} \otimes E_{BC}} \otimes E_{UV} - \hat{N}_{UV} \otimes E_{UV} \\
u &= \frac{(V - B) \otimes S}{R \otimes S} \\
X &= B + u \cdot R' \\
X &= B + \frac{(V - B) \otimes S}{R \otimes S} R' \\
X &= B + \frac{(V - B) \otimes E_{UV}}{R \otimes S} R'
\end{aligned}
$$

The first part of the solution is the same as for the edge crash, but the second part is new, to compensate for the changed bisector ray. By applying the identity

$$
A \otimes \hat{N} = \frac{A \cdot E}{||E||}
$$

the equation becomes

$$
\begin{aligned}
R \otimes S &= \frac{E_{BC}||E_{AB}|| - E_{AB}||E_{BC}||}{E_{AB} \otimes E_{BC}} \otimes E_{UV} + \frac{E_{UV} \cdot E_{UV}}{||E_{UV}||} \\
R \otimes S &= \frac{E_{BC}||E_{AB}|| - E_{AB}||E_{BC}||}{E_{AB} \otimes E_{BC}} \otimes E_{UV} + ||E_{UV}||
\end{aligned}
$$

The equation becomes rather complex, and care must be taken to reduce it. Division should be avoided wherever possible to avoid rounding, and it seems prudent to execute it only at the last step. Similarly, the vector length operations need also to be further cancelled and

rearranged.

$$\begin{aligned} R' &= \frac{D}{d} \\ X &= B + \frac{(V-B)\otimes E}{\frac{D}{d}\otimes E + ||E||}\frac{D}{d} \\ X &= B + D\frac{(V-B)\otimes E}{D\otimes E + d||E||} \end{aligned}$$

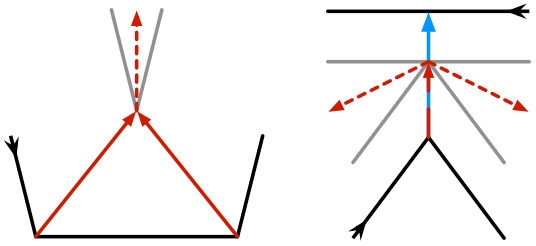
3.7 Dealing with Rounding Errors

From the formula for bisector intersections, it can be seen that due to taking length of a vector, and the involved square root, no absolutely exact intersection time can be computed, even with arbitrary precision decimals. However, the intersection times can be computed exactly to arbitrary precision.

Thus, the question is, what precision is necessary? The fixed point space has a Q16 precision. Attaining this spatial resolution for the intersection point is straight forward. However, for the order of events, the timing is important. How precise does the computed time have to be, to guarantee the right order of events?

3.8 Degenerate and Compound Events

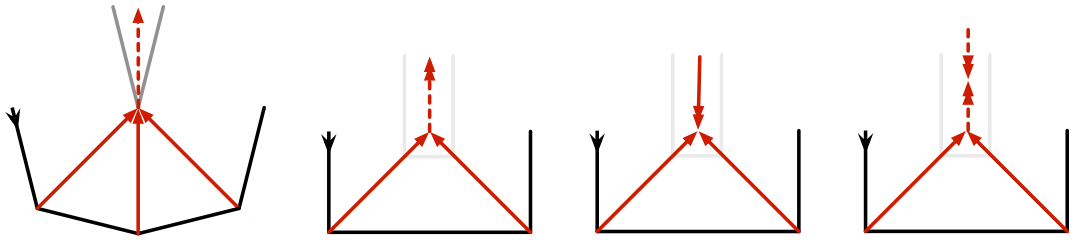
3.8.1 Simple Events



(a) Collapse (b) Split

Figure 1: Simple Events.

Simple events, such as in Figure 1, are no problem to compute. There is no ambiguity or race condition.



(a) Simultaneous Collapse (b) Parallel Neighbours (c) Parallel Neighbours Reverse (d) Degenerate Parallel Collapse

Figure 2: Degnerate Events.

3.8.2 Simple Degenerate Cases

Next up are degenerate events. Figure 2(a) shows a form where multiple wavefronts collapse simultaneously in a single point. This kind of event poses no problem, however, as the order of event resolution does not matter. Similarly, Figure 2(b) is not a problem in itself. The collapse generates a new spoke with infinite velocity. It's reverse condition in Figure 2(c) results in the same situation as Figure 2(a), only that the wavefront propagation terminates. A problem can occur, when the situation in Figure 2(d) arises. If the collapses were computed with perfect accuracy, this would be the actual situation, not the two previous cases. Mathematically the two spokes would collide. However, they clearly do have to collide.

The question arises, how a fast[6] spoke should be handled. The fundamental question is, are fast spokes always generated in pairs? If that is the case, the condition can be detected easily, as the two involved wavefronts form a two-member loop.

3.8.3 Compound Split and Collapse

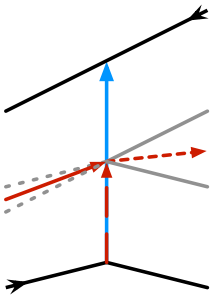


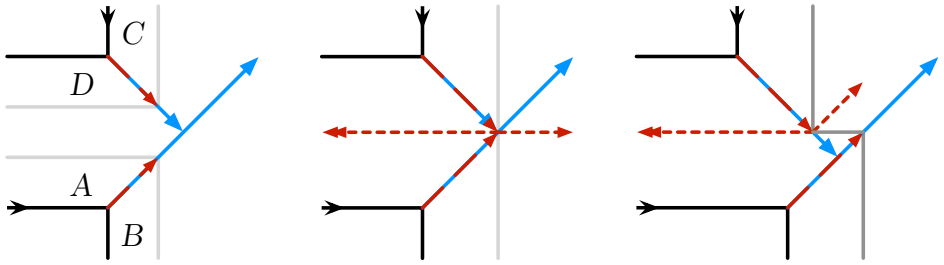
Figure 3: Compound Split and Collapse.

There exist configurations, such as in Figure 3, which pose timing problems between split

[6] an angular bisector with infinite speed

and collapse events. In the figure, the left side of the split collapses simultaneously. Due to limited numeric precision, the split or collapse might occur first. If the collapse occurs first, the split should be annulled, and the collapse resolves to the right configuration.

3.8.4 Terminal Split



(a) Terminal Split Theory (b) Terminal Split Ideal (c) Terminal Split Actual

Figure 4: Terminal Split Edge Case.

Figure 4(a) poses another challenge. Intuitively, the split should occur so that Figure 4(b) is the result. However, the wavefront graph is such that bisector CD will split wavefront A, thus resulting in a situation as in Figure 4(c), with a leftover fragment of A, potentially of infinite length, if AB and CD meet where the split would occur.

If the leftover fragment of A is of finite size, the geometry is not degenerate, though unintuitive. However, an infinitely small wavefront with parallel spokes poses a problem

Some important questions are:

Can the event location be such that wavefronts C and B are oppositely ordered to Figure 4(c)? This case should not occur, as if CD reaches the meeting point before AB, then motorcycle AB would terminate, and the roles of the two motorcycles would be reversed, thus resulting in the same configuration, only mirrored vertically.

The intersection point of AB, CD may be exactly in between grid cells, in which case edges A, D will not be in the same grid cell along CD at any point in time. In this case, the point closer to the origin of CD should be chosen as the split location, as that will always result in a configuration such as Figure 4(c).

Is a zero-length leftover segment of A a problem in further wave propagation? This question assumes that the situation is exactly as in Figure 4(a), with AB and CD reaching their intersection points at exactly the same time.

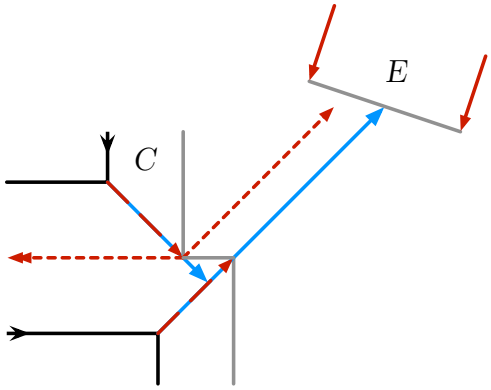
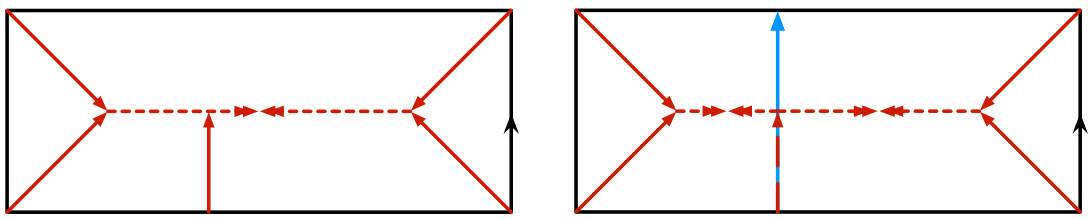


Figure 5: Resolution of split following terminal split.

If the split and terminal vertices are exactly coincident, there is a conflict with the situation as laid out in Section 3.8.3. As shown in Figure 5, the collapse of wavefront C should normally occur after the edge E is split. If both coincide, however, the condition that a split follows a collapse at a given location and time, must be refined.

In the case of coincidence, it seems reasonable to discard the vestigial wavefront immediately, when evaluating the original split event. Coincidence in this case is defined by exactly coincident virtual source vertices for the two spokes, as well as their colinearity.

3.8.5 Parallel Non-Split



(a) Problem (b) Solution

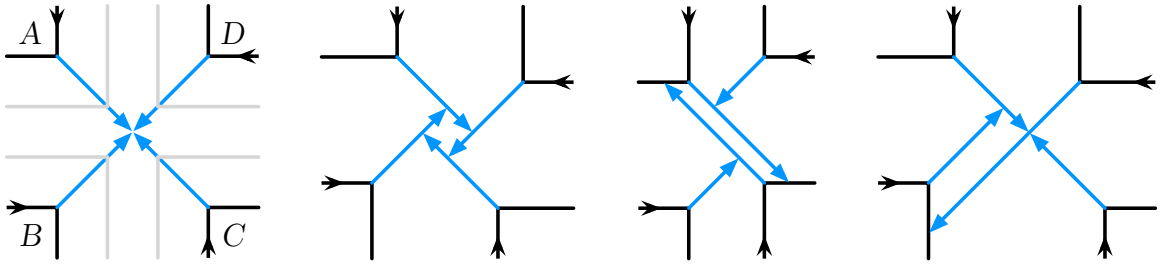
Figure 6: Degenerate Split.

Figure 6 depicts a situation in which the bottom edge is made up of two parts which are not strictly concave, but exactly parallel. This case illustrates that a motorcycle must be generated.

However, the motorcycle only solves the problem that a split event is generated, at all. The wavefronts may only collapse if a proper loop is formed by the two fast spokes. If between the two spokes, there's only two unique wavefronts, then no splitting spoke exists, and they may collapse.

There exists also another problem: if the problematic spoke is not quite a fast spoke, a regular collapse might occur before the split. The proposal that while a wavefront has opposing motorcycles, it may not collapse, not does not work. For *direct* motorcycle-edge crashes, it is fine, but when taking motorcycle-trace crashes into account, the scheme no longer works.

3.8.6 The Degenerate Cross



(a) Ambiguous Cross (b) Wheel (c) Z (d) Asymmetric

Figure 7: Degenerate Cross Motorcycle Graph.

The degenerate cross configuration shown in Figure 7(a) is already non-deterministic at the motorcycle tracing phase, possibly leading to problems later on. All four motorcycles reach the intersection point at the same time, and several different arrangements, as depicted in Figures 7(b), 7(c) and 7(d), can arise, depending on the starting vertex of the source polygon[7], even without the depicted displacement of the four vertices.

Figure 7(d) has been observed to pose a problem, with the result that the split between corners D, A (or C, D) is not completed right, when the four corners are in perfect alignment.

Observing the arrangement, the configuration of D seems straight forward, with a clear opposing front to the spoke. Following this, A and C may be assigned to either opposing or neighbour wavefronts, but the opposing wavefront for B is tricky. The opposing front of A should be D's right neighbour, and B's opposing front should be A's right neighbour.

However,

3.9 Time and Order of Events

The order of events is important when dealing with degenerate cases of wavefront events.

We know that the wavefronts move at a speed of one unit of length per unit of time. Thus, it takes one quantum of time to move one quantum of space.

[7] that is the assumption, at least

The tip of a spoke, as a point, describes the moving intersection point of the two wavefronts, as they are moved along their normals. When a wavefront collapses, the resulting spoke can be tracked from the virtual intersection point of the original source edges, resulting in no accumulating error in the geometry over successive collapses. On a split, a spoke can be tracked in the same manner.

Assuming that the event locations tracked in this manner are accurate to the spatial quantum, it follows that all events at a given point must happen at the same time. With an exact computation, this would be the exact same time. Thus, we can assume that events in the same place also happen at the same time, provided no events occur before.

How do we know that an event happens before a given spoke reaches our event location quantum? It's easy: any location is reached by the spoke either before, or after reaching the location in question. It is not necessary to compare with another spoke, thus compound rounding errors are not an issue. So for comparing the time on any two events for a given spoke, it is only necessary to compute time to the location quantum, and this comparison only has to be made with enough precision that the time to one location is distinct from the next.

The remaining caveat are spokes with infinite velocity. Practically, they pose no problem. Their source segments cannot be intersected, so the intersection point must be computed differently. The intermediate point between any two points on the source segments is well enough.

3.9.1 Event Time Definition

The time of an event is determined by when the last involved wavefront reaches the event location. The square of the time can be computed to high precision. This time will differ from the actual event time, but is a more reliable measure for event ordering[8].

3.9.2 Coalescing Events

An interesting degenerate case occurs when we have three potential events, A, B, C. A, C have the same location, B has a different location. The events B, C involve the same spoke. The order of events is $A < B < C$. By the above definition, A, C belong together, but in this situation, A has a dependency on the later event B.

The only sane way of dealing with such situations seems to be to keep track of events on each spoke, and only allow the first event into the event list.

3.9.3 Relative Timing by Distance Along Spoke

For opposing wavefronts, situations might occur where the order of split and collapse events becomes ill-defined as a result of numerical precision and the space quantization.

[8] why? needs proof.

In the case with test object 3.2, a collapse on one side of the spoke is timed to be before the collapse on the other side, though the two collapses should occur at the exact same time. Without further ordering, the logically wrong collapse can be executed first, resulting in erroneous geometry.

Thus, if two simultaneous collapse events should occur, with differing locations, and they share a common spoke, the one closer to the source vertex should be chosen.

3.9.4 Non-Local Interaction

Some events may need to be suppressed non-locally, and exactly how to do it is yet unanswered.

3.10 Idea: An Event Dependency Graph

As discussed above, the time of events is not enough to sort and determine which can be processed first. A smarter compare function does not seem to work, either, as it is very difficult to make the compares stable. However, maybe it is possible to generate a dependency graph to help with sorting.

For example, a split will depend on either neighbouring wavefront of the spear collapsing, or the opposing wavefront collapsing. However, the opposing wavefront being split by another spear would not affect this split, it would only result in rearrangement of the wavefront graph, without altering timing of this event.

3.11 Exactness

All events during polygon offsetting result from angular bisectors, and the intersections of the angular bisectors and the segments of the source polygon.

3.11.1 Conclusion

In conclusion, ???.

4 Boolean Operations on Polygons

Boolean operations on polygons are necessary for several FDM features.

4.1 Simple Polygons with Holes

The polygons we're dealing with are simple, in the sense that they are non self-intersecting.

References

[1] P. Felkel and Š. Obdržálek, "Straight skeleton implementation," in *SCCG 98: Proceedings of the 14th Spring Conference on Computer Graphics, Budmerice, Slovakia* (L. Szirmay-Kalos, ed.), pp. 210–218, Comenius University, Bratislava, 1998.

[2] T. Hunter, "Gantry support howto."

[3] Moldflow Plastics Labs, "Moldflow material testing report mat2238."

[4] E. W. Weisstein, "Line-line intersection."

[5] J. R. Shewchuk, "Adaptive precision Floating-Point arithmetic and fast robust geometric predicates," in *Discrete and Computational Geometry*, vol. 18, pp. 305–363, 1997.

[6] J.-H. Haunert and M. Sester, "Area collapse and road centerlines based on straight skeletons," *GeoInformatica*, vol. 12, no. 2, pp. 169–191, 2008.

[7] M. Tanase and R. C. Veltkamp, "A straight skeleton approximating the medial axis," tech. rep., Institute of Information & Computing Sciences, Utrecht University, The Netherlands, 2004.

[8] L. Dinu, "Straight sekeleton," 2009.

[9] L. Yan, "Algorithms for computing motorcycle graphs," Master's thesis, King Abdullah University of Science and Technology, 12 2012.

[10] S.-W. Cheng and A. Vigneron, "Motorcycle graphs and straight skeletons," tech. rep., Dept. of Computer Science, Hong Kong University of Science & Technology, 11 2001.

[11] J. Erickson, "Crashing motorcycles efficiently," 02 1998.

[12] S. Huber and M. Held, "An approach to computing motorcycle graphs," 03 2009.