Foxtrot 提交演进报告（sboxgen run 汇总）

2025 年 9 月 9 日

目录

1 提交考古：001-75ebbb7

Commit 元信息

- 标题：First commit
- 作者：Matt Keeter <matt.j.keeter@gmail.com>
- 日期：2021-04-03T15:57:59-04:00

变更摘要（阅读提示）

```
HEAD:   5 files changed, 77 insertions(+)
        + .gitignore, Cargo.lock, Cargo.toml, src/lib.rs, src/main.rs
HEAD-1: N/A（初始提交）
HEAD-2: N/A（初始提交）
```

差异解读（证据）

- 改了什么：新增 Rust 工程骨架（Cargo.toml/Cargo.lock），在 `src/lib.rs` 实现 `circumdelta/circumcenter/circumradius2`,并将 `geometry_predicates::incircle` 以 snake_case 形式重导出为 `in_circle`（见 `HEAD.diff`: src/lib.rs）。
- 为什么改：以 Shewchuk 几何谓词为理论依据，采用相对坐标计算外接圆心以提升数值稳定性与精度；引入 `geometry-predicates` 作为鲁棒几何基础（见 `Cargo.toml` 新增依赖）。
- 影响何在：为后续 CDT/三角化等算法提供基础几何运算；构建层面新增第三方依赖；在退化三点共线（`orient2d=0`）分母为零存在风险，需后续加保护或早期判定（见 `circumdelta` 中对 `orient2d` 的使用）。
- 如何验证：在 `head/` 快照内执行 `cargo build --release`；用已知三角形（如直角三角形）比对 `circumcenter` 与理论值；对共线点集验证异常路径（应返回错误/NaN 防护）。

演进脉络

- head-2 → head-1：N/A（无更早提交）。
- head-1 → head：N/A（初始即引入核心几何功能与依赖）。

证据摘录（文件 + 行区间）

- src/lib.rs:L1–L49：实现 `circumdelta/circumcenter/circumradius2`, 基于 `orient2d` 计算外接圆心与半径平方。
- src/lib.rs:L45–L49：`pub use geometry_predicates::incircle as in_circle;` 提供对 `incircle` 的 snake_case 接口。
- Cargo.toml:L1–L10：新增工程清单与依赖 `geometry-predicates = "0.3.0"`。

基础知识补充（计算几何）

读取《计算几何教材.md》并结合 Shewchuk 谓词：`orient2d` 的符号判定点集有向面积，可稳定区分左/右转与共线情形；外接圆心可用相对坐标法由边向量与 `orient2d` 构造，数值精度好于直接绝对坐标公式。`incircle` 判定点在三角形外接圆内外，常用于 Delaunay 判据。在本提交中，`circumdelta` 以 `orient2d` 为分母，需对近共线场景加容错，以避免除零或放大误差。

图示与说明（必选）

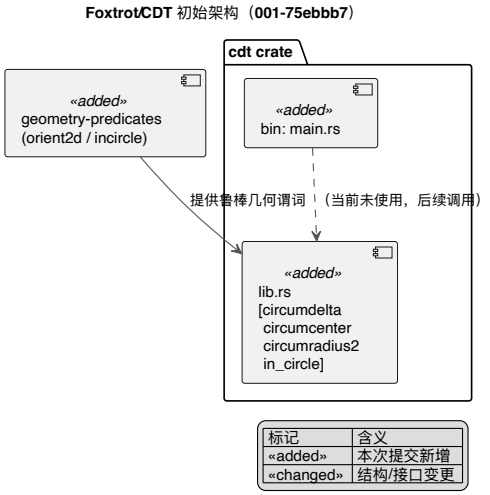

图 1: 架构变化（初始引入 lib/依赖与几何谓词链路）

架构图说明：

- 变化范围：新增 `cdt` crate 与 `lib.rs`，引入第三方几何谓词依赖；
- 依赖与数据路径：`lib.rs` 通过 `geometry-predicates` 获取 `orient2d/incircle` 能力；
- 证据：`HEAD.diff` 中 `Cargo.toml` 与 `src/lib.rs` 新增；
- 风险：依赖带来构建链路变化；退化几何需专门处理以保证鲁棒性。

算法流程说明：

- 入口与阶段：输入三点 a, b, c；计算相对坐标与边长平方；
- 分支与边界：`orient2d(a,b,c)` 近零时视为共线，需回退策略；
- 复杂度与终止：常数时间计算；
- 证据：`src/lib.rs` 中 `circumdelta/circumcenter/circumradius2` 的实现与注释。

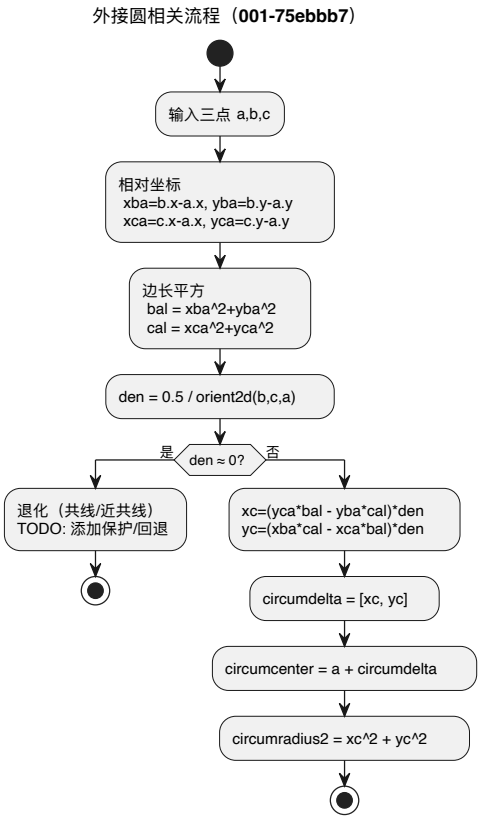

图 2: 算法流程（外接圆心/半径平方与退化判定）

2 提交考古：002-bbb0263

Commit 元信息

- 标题：Add pseudo_angle function
- 作者：Matt Keeter <matt.j.keeter@gmail.com>
- 日期：2021-04-03T16:00:06-04:00

变更摘要（阅读提示）

- diffstat：‘src/lib.rs | 10 ++++++++++‘（新增 10 行）。
- 核心：新增 `pseudo_angle(Point) -> f64`，以分段线性映射在 `[0,1]` 上近似极角，避免三角函数。
- 兼容性：仅新增函数，既有 `circumdelta/circumcenter/circumradius2` 与 `in_circle` 不变。

差异解读（证据）

改了什么

- ‘src/lib.rs‘：在约第 45–59 行新增 `pseudo_angle`（见 `HEAD.diff` 的 hunk ‘@@ -45,5 +45,15 @@‘）。

为什么改

- 性能：极角排序等场景可用伪角代替 `atan2`，避免昂贵三角函数调用。
- 稳定性：使用 `p = x / (|x| + |y|)` 与 `y` 的符号分支构造单调映射，覆盖 0–2π。

影响何在

- API：新增纯函数，无破坏性变更；适用于基于方向的排序、扇区分组等。
- 语义：输出为伪角（归一化至 `[0,1]`），不等同真实角度；精确计算仍应使用 `atan2` 等。
- 边界：输入 `(0,0)` 时分母为 0，上层需过滤或特殊处理。

如何验证

- 基本向量映射：`(1,0)`、`(0,1)`、`(-1,0)`、`(0,-1)` 映射到四等分点附近（允许近似）。
- 单调性：同象限内排序与 `atan2` 一致（以 `atan2` 排序为基准进行对比）。
- 异常输入：`(0,0)` 的处理路径符合上层策略（过滤或约定返回值）。

图示与说明

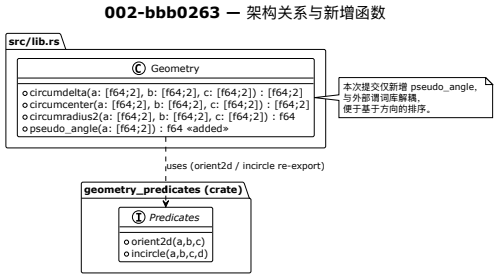

图 3: 模块关系与新增函数：在 lib.rs 中加入 `pseudo_angle`，与现有几何谓词解耦

基础知识补充（计算几何， 200 字）

伪角（pseudo-angle）是一种将二维向量方向单调映射到区间 `[0,1]` 的技巧，常用于基于极角的排序或桶分组，以避免 `atan2` 带来的性能与数值问题。其典型形式以 L1 归一化 `p = x / (|x| + |y|)` 结合 `y` 的正负确定所在半平面，然后线性拼接得到全域单调映射。相较真实角度，伪角保证顺序一致但不保证量值精确，适用在只需“方向次序”的算法（如

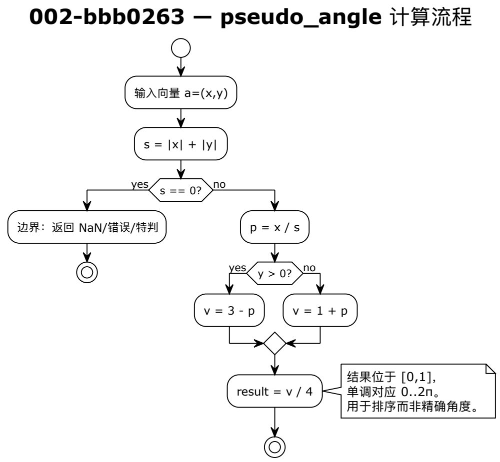

图 4: `pseudo_angle` 计算流程与分支：L1 归一化 + `y` 符号分段

扇区扫描、局部扇形检索等)。本提交新增函数与此一致，并在 `HEAD.diff` 的新增 hunk 中给出实现细节。

3 提交考古：003-8765e49

Commit 元信息

- 标题：Add calculation of initial triangle
- 作者：Matt Keeter <matt.j.keeter@gmail.com>
- 日期：2021-04-03T17:03:06-04:00

变更摘要（阅读提示）

```
HEAD:    Cargo.lock | 41 +++++++++++++++++++++++++++++++++++++++++
         Cargo.toml |  2 ++
         src/lib.rs | 64 +++++++++++++++++++++++++++++++++++++++++++++++++++++-----------
         3 files changed, 96 insertions(+), 11 deletions(-)
HEAD-1:  src/lib.rs | 10 ++++++++++
         1 file changed, 10 insertions(+)
HEAD-2:  First commit (project scaffolding + geometry utilities)
```

差异解读（证据）

- 改了什么：在 src/lib.rs 中引入 triangulate 的“初始三角形”选取逻辑（L71–L101），新增 distance2（L62–L66），并将 Point 从数组改为二元组以便与迭代/排序接口更好配合（L4）。同时在 Cargo.toml 增加 itertools 与 ordered-float 依赖（L11–L12）。
- 为什么改：为增量三角化（Delaunay/CDT）准备稳健的起始三角形，基于几何中心与最短距离/最小外接圆半径的启发式，提升数值稳定性与可维护性；依赖用于便捷地求最值与带序比较键。
- 影响何在：调用路径新增对 itertools::minmax、position_min_by_key 与 OrderedFloat 的依赖；orient2d 调用改为传入坐标数组（L29–L33）以维持几何谓词接口；后续算法将以该三点作为种子并按距外接圆心的距离排序（L97–L100），若输入退化（如少于三点或共线）需要边界处理（当前 expect）。
- 如何验证：最小化样例点集运行 triangulate，检查 a/b/c 选取是否符合“最近中心/最近邻/最小外接圆半径”；对共线/重复点集验证 expect 触发；对比外接圆心与半径的数值一致性（circumcenter/circumradius2）。

演进脉络

- head-2 (初始) → head-1：补充 pseudo_angle（近似角度，无三角函数），为后续排序/扫描等流程提供轻量比较尺度（见 HEAD-1.diff）。
- head-1 (工具) → head：切换 Point 表示、接入 itertools/ordered-float，实现初始三角形选择逻辑并按外接圆心距离排序，为后续增量三角化做铺垫（见 HEAD.diff）。

证据摘录（文件 + 行区间）

- head/src/lib.rs:L71–L101：triangulate 中 a/b/c 的选择与点排序；核心启发式实现与后续流程种子设定。
- head/src/lib.rs:L4 与 L29–L33：Point 改为二元组，orient2d 改以数组实参，接口与数据结构适配细节。
- head/Cargo.toml:L11–L12：新增 itertools、ordered-float 依赖，对应代码中最值/排序键的使用证据。

基础知识补充（计算几何）

关键在于稳健选取 Delaunay 增量构造的起始三角形。教材对 Delaunay 的判定等价性给出“外接圆不含他点”的准则，并以 `incircle`/`orient2d` 等谓词保障数值鲁棒（见《计算几何教材.md》13.8 节与相关讨论）。本提交以“近中心、最近邻、最小外接圆半径”三步启发式挑选 a/b/c，并计算外接圆心用于后续排序，契合“优先避免瘦长三角形”的实践经验；相关实现靠 `circumcenter`/`circumradius2` 与谓词接口支撑（证据见上）。

图示与说明（必选）

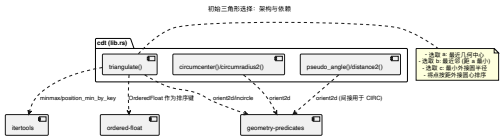

图 5: 架构变化：几何谓词与初始三角形选择的依赖关系

架构图说明：

- 变化范围：`lib.rs` 增加 `triangulate`/`distance2`，`Point` 重定义；
- 依赖与数据路径：`lib.rs` 依赖 `geometry-predicates`（`orient2d`/`incircle`），并引入 `itertools`/`ordered-float` 以实现最值与有序键；
- 证据：`head/src/lib.rs:L71–L101`、`L29–L33`；`head/Cargo.toml:L11–L12`；
- 风险：退化输入与数值边界需处理（当前为 `expect`）。

算法流程说明：

- 入口与阶段：计算包围盒中心 *center*，依次选 a/b/c 并得外接圆心；
- 分支与边界：当点数不足或共线时可能触发 `expect`；`orient2d`/`incircle` 提供鲁棒判断；
- 复杂度与终止：三次全局最值选择与一次全量排序，整体 $O(n \log n)$ 以内；
- 证据：实现见 `head/src/lib.rs:L71–L101` 与相关函数调用。

4 提交考古：004-692603c

Commit 元信息

- 标题：Split into files
- 作者：Matt Keeter <matt.j.keeter@gmail.com>
- 日期：2021-04-03T17:14:40-04:00

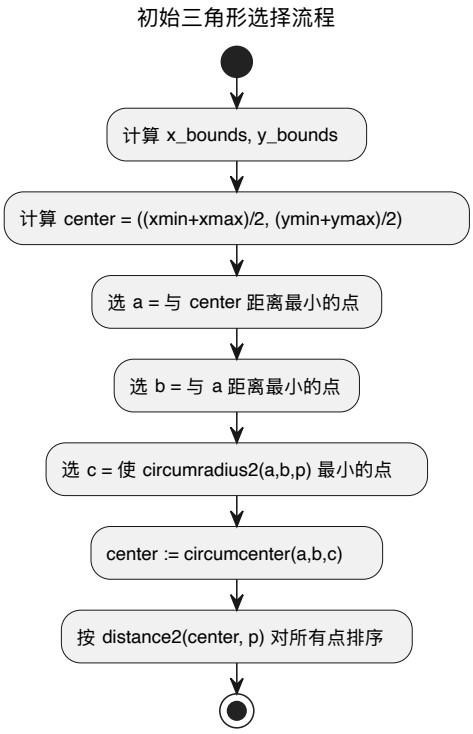

图 6: 算法流程：初始三角形选择与点排序

变更摘要（阅读提示）

HEAD.diff 开头显示：src/lib.rs 大幅精简并改为导出模块，新增 src/predicates.rs 与 src/triangulate.rs；共 3 files changed, 121 insertions(+), 99 deletions(-)。改动核心是将几何谓词与三角化逻辑从单文件拆分为独立模块，便于后续扩展与维护。

差异解读（证据）

改了什么

- src/lib.rs：第 1–4 行改为 ‘pub mod predicates;‘、‘pub mod triangulate;‘（原有函数整体迁移；参见 HEAD.diff: src/lib.rs 1–101 → 1–4）。
- 新增 src/predicates.rs：定义 `circumdelta`/`circumcenter`/`circumradius2`/`pseudo_angle`/`distance` 等（HEAD.diff: src/predicates.rs 1–70）。
- 新增 src/triangulate.rs：引入 `seed_triangle`（选择起始三角形）与 `triangulate` 桩实现（HEAD.diff: src/triangulate.rs 1–49）。
- 依赖：在上一提交添加 `itertools` 与 `ordered-float`（HEAD-1.diff: Cargo.toml 8–12）。

为什么改

- **可维护性**：将几何谓词与三角化流程拆分模块，降低 `lib.rs` 复杂度，形成清晰边界（predicates 专注数值几何，triangulate 承载算法流程）。
- **扩展性**：为引入更完整的 Delaunay/CDT 实现与鲁棒处理（如 `orient2d/incircle`）打基础，便于单元测试与替换实现。

影响何在

- **接口结构**：上层改为从 `predicates`、`triangulate` 命名空间引用函数；构建与模块路径随之调整。
- **算法骨架**：新增 `seed_triangle` 基于质心附近点选择（最近中心、最近相邻、最小外接圆半径）并用 `orient2d` 调整点序；`triangulate` 后续将按距离中心排序推进。
- **风险点**：当前 `triangulate` 未完成；对少于 3 点与共线输入的鲁棒性标注 TODO（HEAD.diff: src/triangulate.rs 注释）。

如何验证

- 结构校验：编译通过且模块可见；对 `predicates` 的单元测试（如三点外接圆心/半径）应独立通过。
- 最小样例：构造三点求 `circumcenter`、`circumradius2` 与 `pseudo_angle` 的数值一致性；`seed_triangle` 在随机点集上返回逆时针序且 `orient2d` 为正。

证据摘录

- HEAD.diff: src/lib.rs 1–4 与 1–101（模块化替换，函数整体迁移）。
- HEAD.diff: src/predicates.rs 1–70（几何谓词函数集中定义）。
- HEAD.diff: src/triangulate.rs 1–49（起始三角形选择与排序骨架）。
- HEAD-1.diff: Cargo.toml 8–12（引入 `itertools`、`ordered-float` 依赖支撑选择与排序）。

图示与说明

基础知识补充（ 200 字）

《计算几何教材.md》“Delaunay Triangulation”与增量构造指出：使用几何谓词（orient2d 判断有向面积、incircle 判断空圆性）可确保三角化的正确性与唯一性；实现层面需考虑浮点误差引发的退化（共线/共圆）。本次提交将 orient2d/incircle 相关调用放入 predicates，triangulate 先选逆时针种子三角形并由中心向外推进，为后续 Delaunay/CDT 的鲁棒实现提供清晰边界与可测性。

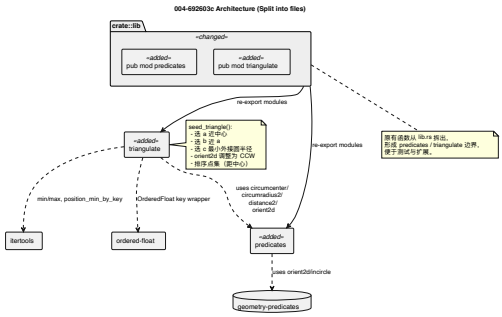

图 7: 模块拆分：lib → predicates/triangulate；依赖与调用路径

5 提交考古：005-eabff6e

Commit 元信息

- 标题：Half-baked Hull
- 作者：Matt Keeter <matt.j.keeter@gmail.com>
- 日期：2021-04-03T18:43:47-04:00

变更摘要（阅读提示）

```
HEAD:   src/hull.rs | 109 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        1 file changed, 109 insertions(+)
HEAD-1: src/lib.rs         | 101 ++-------------------------------------------------
        src/predicates.rs  |  70 ++++++++++++++++++++++++++++++++++++
        src/triangulate.rs |  49 +++++++++++++++++++++++++
        3 files changed, 121 insertions(+), 99 deletions(-)
HEAD-1: Cargo.toml (deps)  | + itertools / ordered-float
```

差异解读（证据）

- 改了什么：新增 `src/hull.rs`（L1–L109），引入基于 `pseudo_angle` 的哈希桶 + 循环链表结构以管理动态凸壳/边界；提供 `insert_initial` 与 `insert` 框架，但实现未完成（半成品）。
- 为什么改：为后续三角化的增量插点（CDT/Delaunay）准备高效 “邻域查找/边界维护” 结构，权衡插入次序与常数因子（使用伪角避免昂贵三角函数，提高性能和鲁棒性）。
- 影响何在：接口层面，点类型统一为 `(f64,f64)`，几何谓词与三角化被拆分为

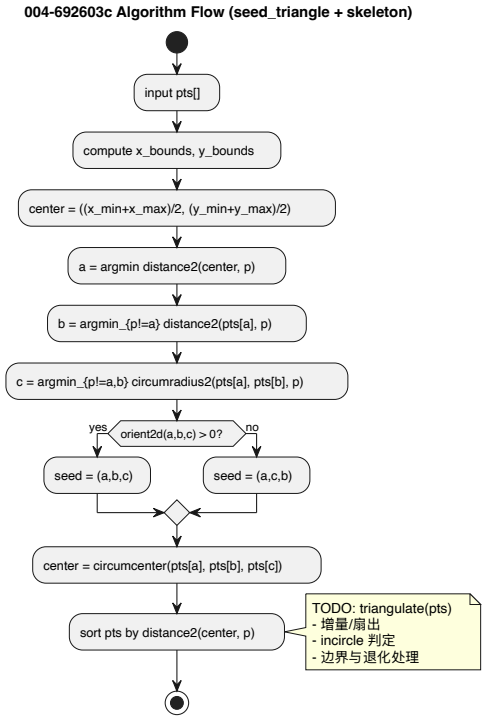

图 8: seed_triangle 与排序骨架；调用 orient2d/circumradius2

predicates.rs / triangulate.rs；构建增加 itertools 与 ordered-float 依赖；边界条件方面，目前 hull.rs 多处占位/变量误用，短期内不可用，需防止误调用。

- 如何验证：最小验证可仅针对 seed_triangle（选择初始三角形）做单元测试；对 hull 结构先用编译检查与静态分析标注 TODO，待补完再引入增量插点测试（含退化/共线样例）。

演进脉络

- head-2 → head-1：抽离几何谓词与三角化逻辑到模块（predicates.rs、triangulate.rs），将点类型由 [f64;2] 改为 (f64,f64)，并引入基于 OrderedFloat 的排序与 itertools 的选择器，完善初始三角形选择流程。
- head-1 → head：新增 hull.rs 半成品，实现哈希分桶 + 伪角排序的外壳结构，作为后续增量三角化的基础设施。

证据摘录（文件 + 行区间）

- head/src/hull.rs:L1--L109（来自 HEAD.diff）：定义 Hull、Key、insert_initial/insert 框架，引入 HASH_SIZE、hints/angles/buckets；但存

在变量名不一致与占位 `unimplemented` 风险。

- `head-1/Cargo.toml:L8-L13`（来自 HEAD-1.diff）：新增依赖 `itertools = "0.10.0"`、`ordered-float = "2.1.1"`，支撑排序与选择逻辑。
- `head-1/src/triangulate.rs:L1--L42`（来自 HEAD-1.diff）：实现 `seed_triangle`：基于边界中心、最近点、最小外接圆半径选择三点，并用 `orient2d` 规范三角形方向。

基础知识补充（计算几何）

伪角（`pseudo_angle`）用 $p = x/(|x|+|y|)$ 将平面向量投射到 $[0,1)$ 区间，无需三角函数且保序性足以用于分桶/排序；`orient2d` 判定有向面积符号，确保三角形一致朝向；`incircle` 可用于后续边翻转/合法性判断。初始三角形通过“中心点最近 → 最近邻 → 最小外接圆半径”三步选择，减少数值退化并提供良好种子（见 `head-1/src/triangulate.rs` 与 `HEAD-1.diff`）。当前 `hull.rs` 仅提供半成品结构，后续可在插点时据 `pseudo_angle` 与哈希桶快速定位候选边界，再配合 `orient2d`/`incircle` 完成合法性维护。

图示与说明（必选）

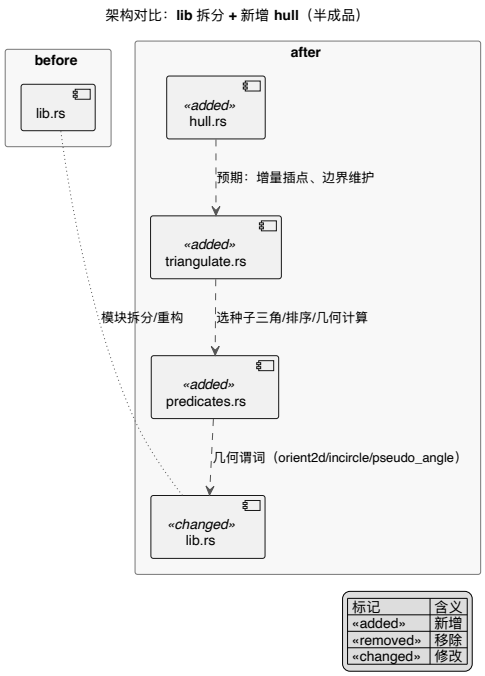

图 9: 架构变化：模块化 predicates/triangulate；新增 hull（半成品）

架构图说明：

- 变化范围：`lib.rs` 拆分为 `predicates.rs`/`triangulate.rs`，并新增 `hull.rs`。

- 依赖与数据路径：引入 `itertools/ordered-float` 以支持排序与选择；三角化依赖几何谓词。
- 证 据：HEAD.diff（`src/hull.rs:+109`） ，HEAD-1.diff（`Cargo.toml:+2`、`triangulate.rs` 新增）。
- 风险：`hull.rs` 未完成，短期禁用；需补充单元测试与边界处理。

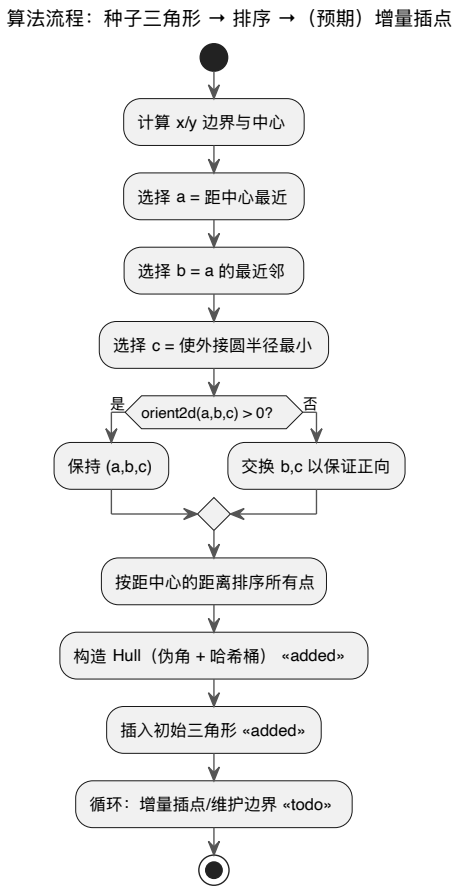

图 10: 算法流程：种子三角形 → 排序 →（预期）增量插点/边界维护

算法流程说明：

- 入口与阶段：计算边界中心 → 选取 A/B/C→ 规范方向 → 按中心距离排序点集。
- 分支与边界：共线/退化需特判；后续增量阶段调用 `orient2d`/`incircle` 做合法性维护。
- 复杂度与终止：种子选择 $O(n)$；增量插点期望 $O(n \log n)$（待实现）。
- 证据：`head-1/src/triangulate.rs` 与 HEAD.diff 的 `hull.rs` 框架。

6 提交考古：006-a0a3b0c

Commit 元信息

- 标题：Move hull into triangulate
- 作者：Matt Keeter <matt.j.keeter@gmail.com>
- 日期：2021-04-03T20:14:22-04:00

变更摘要（阅读提示）

src/hull.rs | 109 deletions（文件被删除）
src/triangulate.rs | 147 insertions/changes（整合 hull 逻辑至 Triangulation）
2 files changed, 107 insertions(+), 149 deletions(-)

差异解读（证据）

改了什么：

- 移除独立 `src/hull.rs`（109 行），将凸包/边结构合入 `src/triangulate.rs`（见新增的 `Triangulation` 与 `Edge`/`EdgeIndex`）。
- 在 `triangulate.rs` 引入 `BTreeMap<OrderedFloat<f64>, EdgeIndex>` 作为 hull 索引，并以 `pseudo_angle` 为键管理外壳边；采用 `NonZeroUsize` 表示有效边索引（保留 #0 为空）。
- 初始化流程集成：`seed_triangle` → 计算外接圆心 → 按距圆心距离排序生成 `order`，并创建三条初始边 e_ab/e_bc/e_ca 填充 hull。

为什么改：

- 维护性：将 hull 数据结构内聚到三角化模块，减少跨文件耦合，便于后续在同一处维护插入/翻边等逻辑。
- 正确性/简单性：以 `BTreeMap` + `OrderedFloat` 替代自制哈希桶与链表，获得稳定的按角有序外壳，降低实现复杂度与潜在错误。

影响何在：

- 调用路径：外部不再依赖 `hull.rs`；相关能力由 `Triangulation` 封装，后续插点/拓扑维护在同一类型内完成。
- 数据结构：新增 `Edge`（含 `buddy` 配对）、`EdgeIndex`（`NonZeroUsize`）、`order`（按外接圆心的距离排序）；外壳以伪角排序，方便旋转/邻接查找。
- 风险：`OrderedFloat` 与伪角在坐标轴附近的数值并列需要关注；后续插点与合法性维护（如 Delaunay 翻边）尚未在本次提交中完善。

证据摘录（来自 HEAD*.diff）：

- HEAD.diff: 删除 src/hull.rs（整文件 109 行）；triangulate.rs 顶部新增 use std::collections::BTreeMap;、use std::num::NonZeroUsize;、use crate::predicates::pseudo_angle 以及 Triangulation 定义与 push_edge。
- HEAD-1.diff: 新增（现已删除）hull.rs，包含哈希桶 + 链表雏形（HASH_SIZE, Key, Hull::new）。
- HEAD-2.diff: 初次分文件；triangulate.rs 引入 seed_triangle 与 predicates 模块（提供 pseudo_angle/orient2d 等）。

如何验证（最小建议）

- 静态检查：确保 triangulate.rs 可编译（类型一致，新增依赖已导入）。
- 行为核对：构造少量点集，检查种子三角形与初始外壳（e_ab/e_bc/e_ca）按伪角顺序入表；后续插点/翻边待后续提交完善。

基础知识补充

伪角（pseudo_angle）以 $x/(|x|+|y|)$ 的有理形式将二维向量映射到 [0,1) 区间，按极角单调，避免开销较大的三角函数，便于构建有序外壳与邻域搜索；在坐标轴附近需注意数值并列与稳定性。

图示与说明

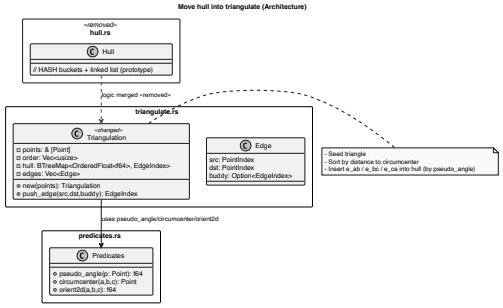

图 11: 架构变化：Hull 合入 Triangulation，外壳以 BTreeMap+ 伪角管理

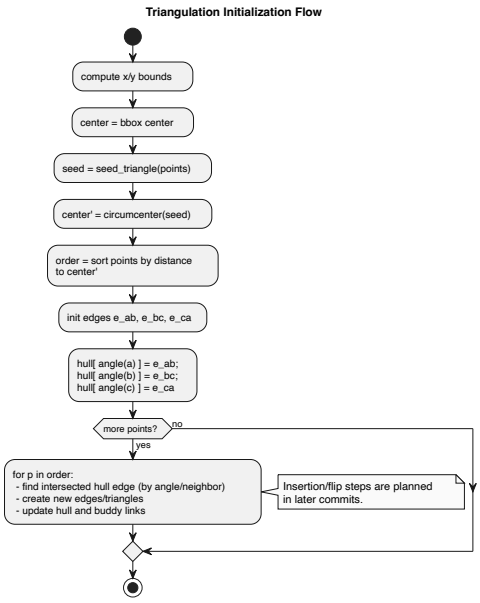

图 12: 初始化流程：种子三角形 → 排序 → 三边入壳；为后续插点奠基

7 提交考古：007-27c3c0a

Commit 元信息

- 标题：Add a basic test
- 作者：Matt Keeter <matt.j.keeter@gmail.com>
- 日期：2021-04-04T09:44:15-04:00

变更摘要（阅读提示）

本次仅修改 `src/triangulate.rs`（65 insertions, 17 deletions）。核心点：为三角化结构加入中心点 `center` 与基于中心的伪角键 `key()`，修正种子三角形选取流程与点序列过滤，并新增基础单测。

差异解读（证据）

改了什么

- 结构体与键：为 `Triangulation` 增加 `center` 字段，并以相对中心的 `pseudo_angle` 作为凸包键（`key()` 新增）。见 `src/triangulate.rs`：新建 `key`（约 L46–L55），`new` 中改为 `out.key(...)` 插入（约 L30–L45）。

- 种子三角形：在 `seed_triangle` 中避免 `a/b` 自身参与比较，使用 `INFINITY` 规避（约 L96–L120）。
- 辅助与测试：新增 `get_hull_edge`（约 L132–L140），并添加 `#[cfg(test)] mod tests` 的基础用例（约 L142–L160）。

为什么改

- 正确性与稳定性：将凸包键从“全局伪角”改为“以初始外接圆心为中心的相对伪角”，可以稳定排序并与后续点的投影查询（`get_hull_edge`）对齐。
- 可测试性：加入最小单测确保点序列过滤后，除种子三点外按到中心距离从内到外排序、并保持预期索引。

影响何在

- 调用路径：`Triangulation::new` 的初始化顺序变化（中心计算 → 排序 → 过滤 → 建立边/凸包）。
- 边界条件：在种子三角形选取时显式排除自身候选，减少退化与数值并列导致的误选风险。
- 结构演进：结合上一提交移除 `hull.rs`，凸包逻辑现内聚于 `triangulate.rs`，以 `BTreeMap<OrderedFloat<f64>, EdgeIndex>` 管理。

如何验证

- 构建与测试：在原项目中 `cargo test` 可运行新增单测；当前仅保留 diff 证据，最小验证参照下文建议。
- 观察序列：断言 `order` 已过滤种子三点且仅包含外点索引（测试中断言长度与具体索引）。

证据摘录

- 伪角键引入与使用：`src/triangulate.rs` 中 `key()` 新增与 `hull.insert(out.key(...))`（见 `HEAD.diff` 中 `@@ -34,38 +34,52 @@` 与后续 `@@`）。
- 种子三角形的候选排除：`seed_triangle` 将与 `a,b` 同索引的候选赋 `INFINITY`，避免参与最小化（`HEAD.diff` 约 `@@ -96,14 +110,20 @@`）。
- 新增 `get_hull_edge` 与测试模块：见 `HEAD.diff` 末段两个新增 hunk（约 L132+ 与 L142+）。

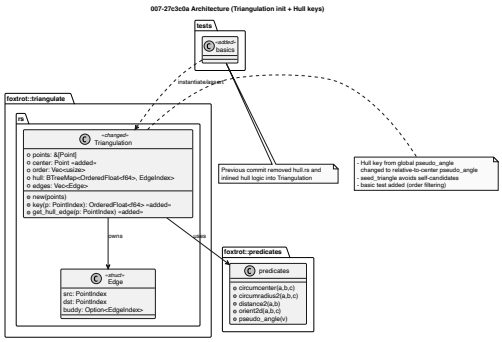

图 13: 架构与数据流变化（凸包键从全局伪角改为相对中心伪角；Hull 内聚于 Triangulation）

图示与说明

基础知识补充（计算几何）

参考《计算几何教材.md》“Delaunay Triangulation”（约第 652 行起）与“鲁棒性与退化情形”（约 2000 行附近）。增量三角剖分常以外接圆心/半径作为度量（`circumcenter`/`circumradius2`），插点时需借助 `orient2d`/`incircle` 保证拓扑正确；构建凸包或有序边界时，采用一致的角度（或伪角）排序可降低数值不稳定导致的错序风险。结合本次改动，以中心为基准的伪角键与候选排除策略，有助于稳定初始化并为后续点投影到边界提供一致性键。

最小验证建议

- 在原仓库：执行 `cargo test --package foxtrot`，应通过新增的 `basics` 测试；或在 `head/` 快照内构建以验证编译通过。
- 额外断言：打印并检查 `order` 与 `hull` 的键顺序，确认与中心相对的伪角单调一致。

8 提交考古：008-4d1a91f

Commit 元信息

- 标题：Begin work on half-edge data structure
- 作者：Matt Keeter <matt.j.keeter@gmail.com>
- 日期：2021-04-04T13:23:26-04:00

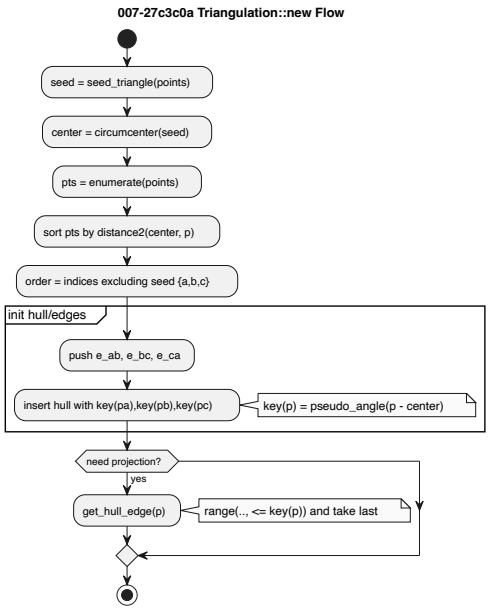

图 14: Triangulation::new 初始化与点序构建流程

变更摘要（阅读提示）

本次提交引入半边（half-edge）数据结构作为三角网格的核心表示，重构 `triangulate.rs` 以使用 `Half` 管理边关系，并新增 `to_svg()` 以输出调试用 SVG。主要改动：新增 `src/half.rs`，在 `src/lib.rs` 中抽出 `PointIndex/EdgeIndex`，`src/main.rs` 用最小示例打印 SVG。

差异解读（证据）

改了什么

- 新增 `src/half.rs`（约 145 行）：定义 `Edge` 与 `Half`，提供 `insert/next/prev/swap/iter_edges`。见 HEAD.diff 段 `src/half.rs @0..145`。
- `src/lib.rs` 抽象索引类型：引入 `PointIndex/TriangleIndex/EdgeIndex`（基于 `NonZeroUsize`），导出 `pub mod half;`。见 HEAD.diff 的 `lib.rs` 变更。
- `src/triangulate.rs`：- `Triangulation` 结构体改用 `Half` 替代手写 `edges` 数组；- 移除本地定义的 `PointIndex/EdgeIndex/Edge`；- 新增 `to_svg()`（见 `@@ +107,+39`）遍历 `half.iter_edges()` 绘制线段；- 初始化时用 `half.insert()` 建立种子三角形并更新 `hull`。见 HEAD.diff 多处 `@@` 标记（例如 `@@ -1,26 +1,13`，`@@ -61,35 +45,17`，`@@ -141,6 +107,39`）。
- `src/main.rs`：以 `Triangulation::new` 构造并 `println!(t.to_svg())` 替代 `Hello,`

world!。

为什么改

- **结构化网格拓扑**：半边结构天然表达三角形邻接与边对（buddy），后续支持边翻转（swap）等操作以靠近 Delaunay/约束 Delaunay 的增量算法需求（见 half.rs::swap）。
- **类型安全与开销控制**：EdgeIndex(NonZeroUsize) 配合保留槽位 0，令 Option<EdgeIndex> 无额外内存开销（lib.rs 注释）。
- **调试可视化**：to_svg() 直接渲染 half 中所有边，便于快速验证构网流程与几何关系。

影响何在

- 调用路径：原先 Triangulation 手动维护 edges 改为委托 Half，遍历与更新均通过 half.* 接口完成（接口面向索引，屏蔽点坐标）。
- 边界条件：swap 要求存在配对边（buddy.is_some()），错误时返回 Err(()) ——后续调用方需判断可翻转性以避免 panic。
- 构建与输出：main 现在输出 SVG 字符串；下游工具链如管道/重定向可直接保存预览图。

如何验证

- 最小样例（同 src/main.rs）：以四点 (0,0),(1.5,0),(0,1),(2,2) 构造 Triangulation 并打印 to_svg()；检查是否含三角形边与绿色标记点。
- Half 邻接：调用 half.iter_edges() 应至少包含种子三角形三条有向边；对具配对的边调用 swap() 后，next/prev/端点应符合菱形翻转示意。
- Hull 顺序：hull 以中心点的 pseudo_angle 排序，点插入前后相邻键应单调有序（见 get_hull_edge）。

关键证据摘录

- src/half.rs 新增文件（核心接口）：insert/iter_edges/swap，见 HEAD.diff 开头新增块。
- src/triangulate.rs to_svg():@@ -141,6 +107,39 起新增,遍历 self.half.iter_edges() 输出 <line/>。
- src/lib.rs：定义 EdgeIndex(NonZeroUsize) 与 Default，并 pub mod half；使 Triangulation 可见半边实现。

基础知识补充（200 字）

增量式 Delaunay/约束 Delaunay 常以“半边”表达拓扑，并通过局部“边翻转（edge flip）”维护空外接圆性质。初始三角形可取中心最近点—最近邻—最小外接圆半径的三点并保证逆时针；随后按相对中心的 `pseudo\_angle`（近似极角，单调映射，便于排序）由内向外插点并更新凸包。`swap` 仅在存在配对边（共用四边形）时可行。以上与 `HEAD*.diff` 中 `half.rs::swap`、`triangulate.rs::to_svg/key/get_hull_edge` 相互印证。

图示

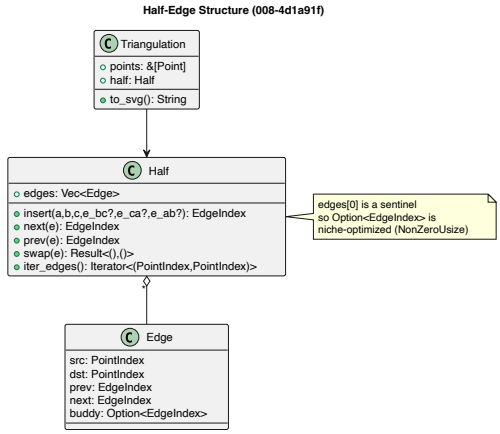

图 15: 半边结构与三角单元关系

9 提交考古：009-763ede8

Commit 元信息

- 标题：SVG tweaks
- 作者：Matt Keeter <matt.j.keeter@gmail.com>
- 日期：2021-04-04T13:26:49-04:00

变更摘要（阅读提示）

本次改动聚焦 `src/triangulate.rs` 的 `to_svg`：将背景矩形由黑色填充改为 `fill:none` 并补充注释，避免 `rsvg-convert` 导出 PDF 时对边线裁剪或遮挡；保留遍历半边输出边线与原点圆（见下小节证据）。

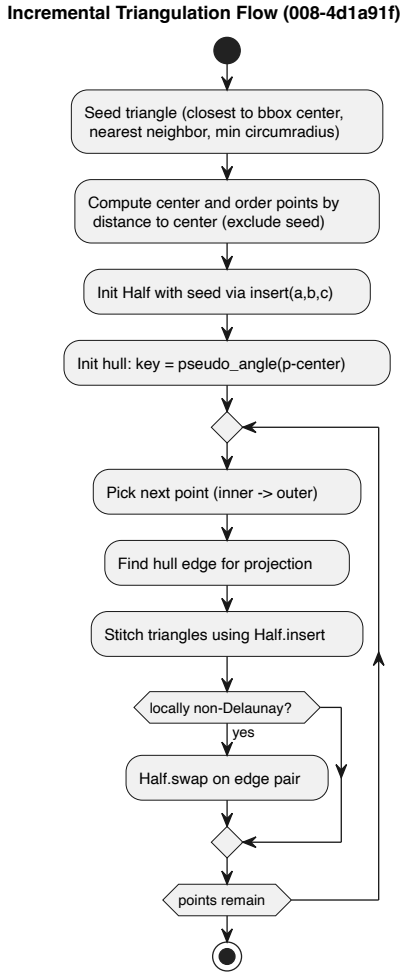

图 16: 增量三角化流程与可视化路径

差异解读（证据）

- 改了什么：
  - `src/triangulate.rs:116--140`（HEAD.diff）：新增注释；矩形样式由`fill:rgb(0,0,0)` 改为 `fill:none`；保留对半边的遍历输出 `<line>` 与原点 `<circle>`。
  - `src/half.rs:1--145`（HEAD-1.diff）：引入 `half::Half` 半边结构（`insert`/`iter_edges`/`swap`）。
  - `src/lib.rs:1--19`（HEAD-1.diff）：加入 `PointIndex`/`EdgeIndex` 定义并公开模块。
- 为什么改：导出 PDF 时若无外包围矩形，转换器可能裁剪超出视窗的几何；设置透明矩形可保持几何边界且不遮挡绘制。

- 影响何在：
  - 渲染稳定性：SVG→PDF 转换更稳定，避免边缘被裁剪；视觉不再被黑底遮挡。
  - 维护性：导出意图与步骤写入注释，后续修改更清晰。
- 如何验证：
  - 运行示例（HEAD-1 的 `main.rs`）打印 SVG；用 `rsvg-convert` 或 `sips` 生成 PDF，观察边线完整性。
  - 如缺少工具：浏览器直接打开 SVG，确认背景透明且边线未被裁剪。

基础知识补充

半边结构以三条有向边隐式存三角形，便于遍历与翻转；`key()` 使用伪角（pseudo-angle）对点按中心极角排序，将凸包边映射到有序表以便增量更新。`to_svg` 仅负责可视化：计算边界与线宽、坐标平移、写入透明占位矩形、遍历半边为红线、在原点绘制圆点，保证导出稳定。

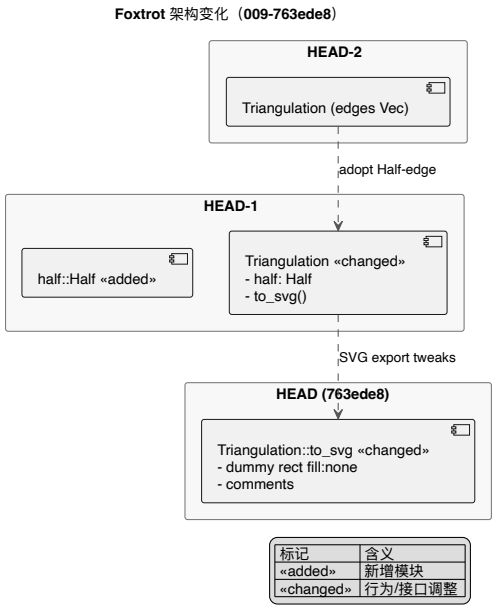

图 17: 架构演进：从 Vec<edge> 到 half::Half，再到 SVG 导出细化

10 提交考古：010-4c76e2e

Commit 元信息

- 标题：Flip XY and cap strokes

- 作者：Matt Keeter <matt.j.keeter@gmail.com>
- 日期：2021-04-04T13:45:06-04:00

变更摘要（阅读提示）

```
HEAD:   src/triangulate.rs | 10 ++++++----
        1 file changed, 6 insertions(+), 4 deletions(-)
HEAD-1: src/triangulate.rs | 7 ++++++-
        1 file changed, 6 insertions(+), 1 deletion(-)
HEAD-2: src/half.rs        | 145 +++++++++++++++++++++++++++++++++++++++++++++++++++++
        src/lib.rs         |  15 ++++++
        src/main.rs        |   8 ++-
        src/triangulate.rs |  86 ++++++++++++++++---------------
        4 files changed, 210 insertions(+), 44 deletions(-)
```

差异解读（证据）

- 改了什么：在 `src/triangulate.rs` 的 `to_svg` 中翻转 Y 轴映射、减小线宽比例并设置圆形线帽（约 `L111–L136`）；同时修正背景矩形高度以避免翻转后裁剪（HEAD）。
- 为什么改：提升 SVG 可读性与一致性。Y 翻转贴合屏幕坐标系（Y 向下）；更细线宽与 `stroke-linecap=round` 改善边的视觉质量；背景矩形避免边缘被裁剪（与 rsvg-convert 兼容）。
- 影响何在：仅影响渲染输出的坐标和线端样式；不影响几何/数据结构。下游若依赖旧的 Y 方向或线宽，截图/比对需更新；SVG 消费侧无需代码变更。
- 如何验证：构造少量点集，调用 `Triangulation::to_svg()`，肉眼比对上下翻转是否正确，线端是否圆角，整体是否无裁剪。可复用 HEAD 2 中的示例 `main.rs` 将 SVG 打印后查看。

演进脉络

- head-2 → head-1：在 `to_svg` 加入“防裁剪”占位矩形并改为 `fill:none`（由原先黑底），补充注释与调试用原点圆点（见 HEAD-1）。
- head-1 → head：翻转 Y 轴（`dy(y)=y_max - y - lw`）、线宽由最大边界的 1/20 改为 1/40，并设置 `stroke-linecap=round`，同时修正高度计算（见 HEAD）。

证据摘录（文件 + 行区间）

- `src/triangulate.rs:L111–L136`（HEAD）：`line_width` 从 `/20` 到 `/40`；`dy` 改为以 `y_max` 翻转；线条样式改为独立属性并新增 `stroke-linecap=round`；矩形高度改为 `dy(y_bounds.0)+2*lw`。
- `src/triangulate.rs:L116–L136`（HEAD-1）：新增防裁剪矩形且 `fill:none`，并添加 “Push every edge” “Add a circle at the origin” 注释和原点圆点。
- `src/half.rs:L1–L145`（HEAD-2）：新增半边结构 `Half` 及边迭代；`Triangulation` 从自管边向半边抽象迁移；首次引入 `to_svg` 输出管线。

基础知识补充（计算几何）

读取《计算几何教材.md》“Delaunay 三角化/增量构造” 相关部分并检索 pseudo-angle：伪角用于对相对中心的向量进行单调排序，避免昂贵的 `atan2`，常用于维护凸包/环形序（见 HEAD-2 在 `key()` 中对 `pseudo_angle` 的调用）。渲染层采用屏幕坐标系（Y 向下），故在 SVG 导出中进行 Y 翻转以匹配显示设备（见 HEAD 对 `dy` 的改动）。

图示与说明（必选）

架构图说明：

- 变化范围：`Triangulation::to_svg()` 渲染层；`Half` 作为边存储与迭代；
- 数据路径：`Half.iter_edges()` → 计算 `dx/dy` → 生成 SVG `<line>/<rect>/<circle>`；
- 证据：HEAD 与 HEAD-2 中 `to_svg` 和 `Half::iter_edges` 实现；
- 风险：仅视觉变化；和导出管线（如 `rsvg-convert`）的裁剪/坐标假设需对齐。

算法流程说明：

- 入口：给定点集与半边结构，计算 `x/y` 边界与 `line_width`；
- 关键：`dy(y)=y_max - y - lw` 完成 Y 翻转；线条设为圆角以减少视觉锯齿；
- 边界处理：占位矩形避免裁剪；
- 终止：遍历完 `iter_edges()` 后输出 `</svg>`；
- 证据：对应 HEAD/HEAD-1/HEAD-2 中的 `to_svg` 片段与注释。

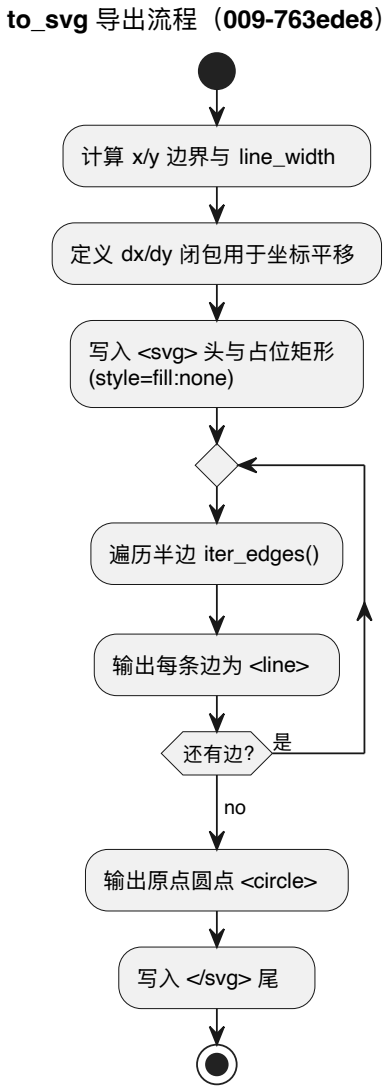

图 18: to_svg 导出流程（含透明占位矩形避免裁剪）

图 19: 架构变化：Triangulation 依赖 Half 提供边迭代，to_svg 负责坐标变换与标注（Y 翻转、圆角线帽）

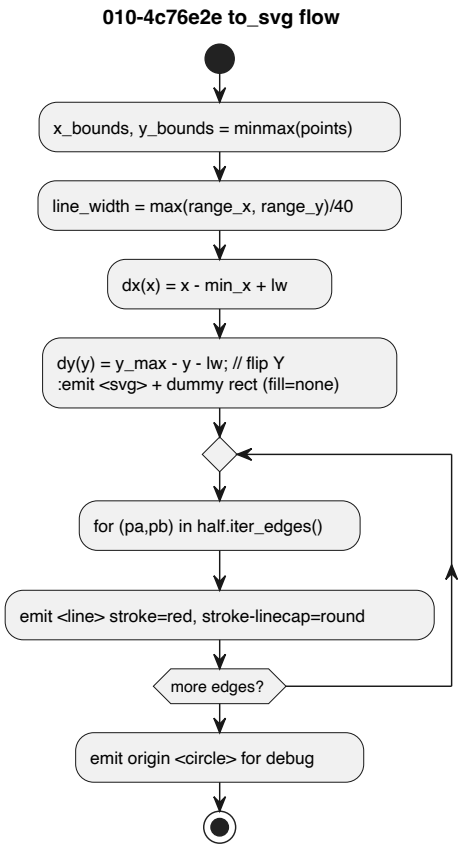

图 20: 算法流程：边界计算 → 线宽与坐标变换 → 占位矩形 → 遍历半边画线 → 原点标记