Описание объемного режима представления топосъемки программы Topo.exe

1. Элементы управления

Иллюстрация 1: Главная панель управления объемного режима

Новая панель управления содежит 5 кнопок:

а) Переключатель режимов съемки

 классический режим — построение нитки хода, доступны все классические возможности программы.

 - режим границ пещеры – в этом режиме строится только граница пещеры и сечения в районе пикетов

 - режим сечений – в этом режиме LRUD и «ежики» каждого пикета визуализируются своей поверхностью

 - режим не сглаженного отображения объемов, включается специальный отрисовщик, недоступен ряд классических функций

 - режим сглаженного отображения объемов, включается специальный отрисовщик, недоступен ряд классических функций, подробнее см. в разделе 5.2

б) Переключатель режима раскраски в объемном режиме

 - градация цвета по габаритам хода, измеряется наименьший линейный размер поперечного сечения:

0.325 м — красный

0.75 м — желтый

1.5м - 5 м — зеленый

11м — голубой

 - градация цвета по глубине (как в терионе), красный — верх топосъемки, фиолетовый — низ топосъемки.

 - цвет берется из данных съемки, задаваемой командой #color[n]

г) Переключатель отображения нитки хода в объемном режиме (- вкл/ - выкл)

д) - Переключатель отображения панели дополнительных функций

е) - Переключатель отображения панели отладочных функций

2. Ввод данных

Формат ввода данных для последующей обрисовки поверхности стен доступен в 3х форматах:

a) LRUD — запись расстояния до стен слева, справа, сверху, снизу на каждом пикете.
   Например:
   #data_order L Az An **l r u d**
   1 2 2.2 95 4.4 **0.25 0.25 1.5 0.5**

b) С помощью команды #walls[n] ^label, где следующие n записей являются замерами до станы от пикета label
   Например:
   #data_order L An Az
   piket1 piket2 2.2 95 4.4
   #walls[4] ^piket1
   1.6 340 50
   2.8 116 34
   1 179 42
   0.9 295 -61

c) Указывая знак прочерк вместо имени второго пикета
   #data_order L Az An
   piket6 piket7 3.2 115 -57
   piket6 - 0.1 10 0
   piket6 - 0.5 190 0
   piket6 - 1.4 280 20
   piket6 - 0.9 105 0.1

Нет никаких ограничений в отношении совместного использования всех трех методов в рамках одной топосъемки. Подробнее о командах — в справке по программе Topo.exe

2. Требования к измерениям до стен

Используемые в программе алгоритмы накладывают ограничения на данные съемки расстояния до стен.

При съемке расстояния до стен нужно стремиться снять **характерное сечение хода в окрестности пикета**. В идеале, через все замеры до стен **должно быть можно провести одну плоскость,** но не обязательно перпендикулярную направлению хода. Однако даже если некоторые замеры будут отстоять на угол до 35-40 градусов от этой воображаемой плоскости, программа все равно корректно их обработает. Страдать качество модели начнет при отклонении больше 45 градусов.

Определенно неправильно прийти на пикет и начать снимать расстояния до стен во все стороны не следуя никакому правилу — по таким данным будет построена рваная, ломаная, явно неправильная картинка. Через некоторое время возможно Topo научится обрабатывать и такие данные. Что вызвало такие ограничения, будет рассмотрено в главе 2.

Примеры:

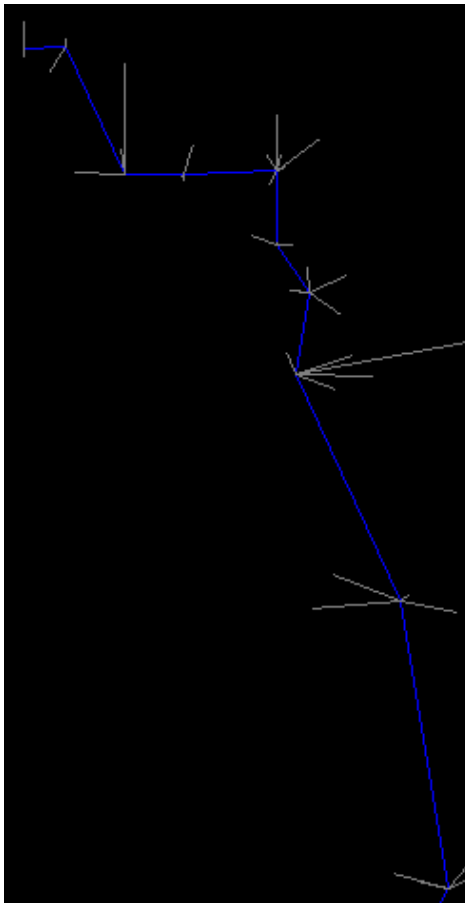


Иллюстрация 3: В большинстве правильные измерения до стен.

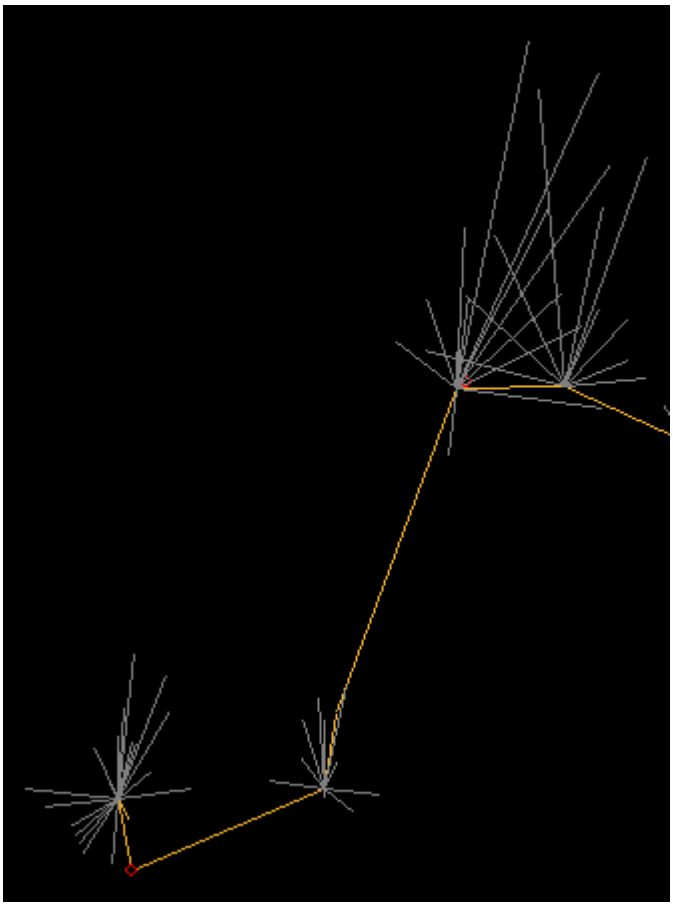


Иллюстрация 2: В большинстве неправильные измерения до стен

3. Рекомендации по съемке

Ниже приведены участки пещеры часто встречающейся морфологии и рекомендации по расположению пикетов и плоскостей сечений в них.

3.1. Относительно прямой монотонный ход.

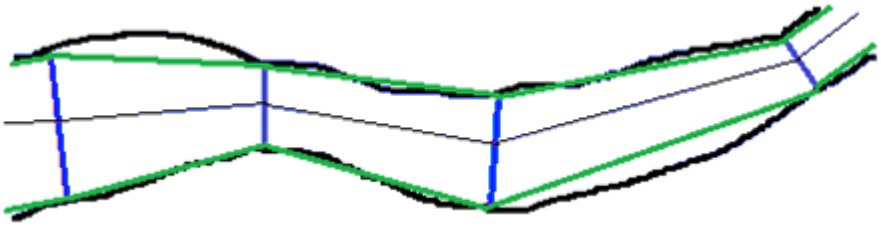


Иллюстрация 4: Съемка череды перпендикулярных ходу сечений

3.2. Тупик

Иллюстрация 5: Для отрисовки тупика в его конце ставится пикет с одним единственным отстрелом (в том числе, нулевой длины)

3.3. Поворот, выход под колодец

Иллюстрация 6: Съемка поворота

3.4. Развилка

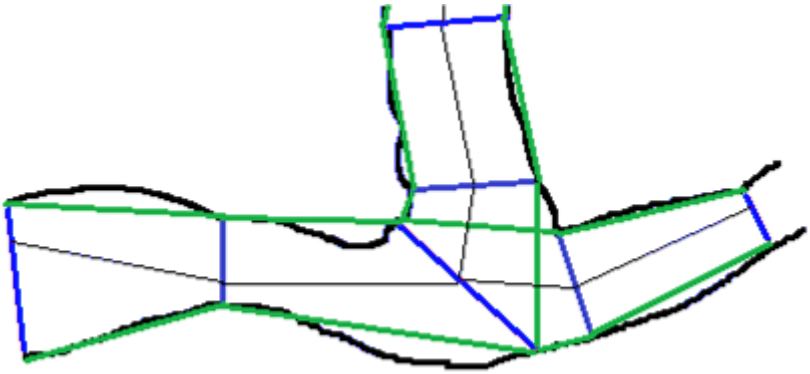


Иллюстрация 7: Съемка развилки - комбинация съемок поворота и прямого хода

3.5. Ход с локальными расширениями - гротами, провалами, приходящими колодцами.

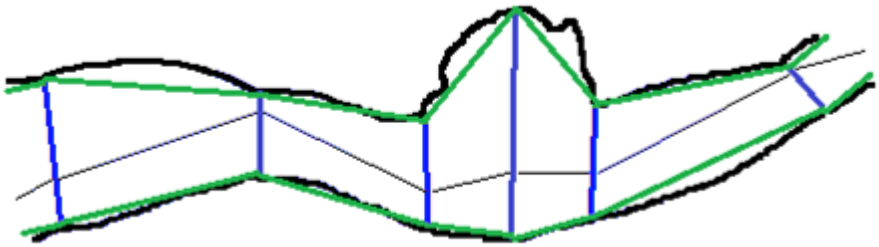


Иллюстрация 8: Вариант первый - один пикет до расширения, один в расширении, один после

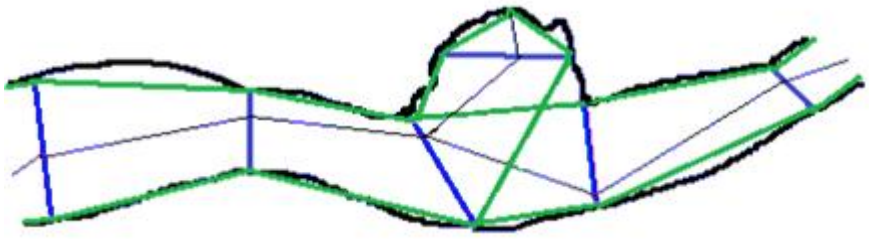


Иллюстрация 9: Вариант второй - через формирование развилки и тупика

3.6. Разворот

Выход в колодец, а также приход сверху неисследованных колодцев в галереях и залах

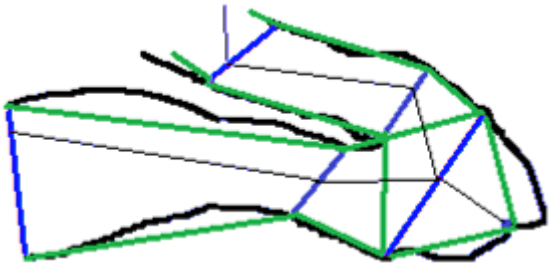
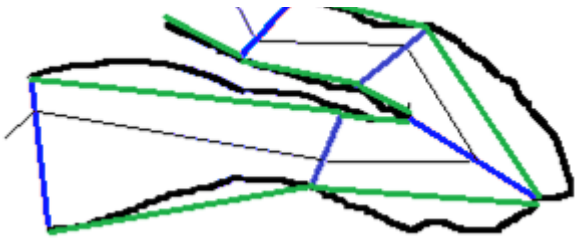


Иллюстрация 10: Второй способ — снять как развилку и тупик



Первый способ - снять разворот неразрывно

3.8. Съемка залов

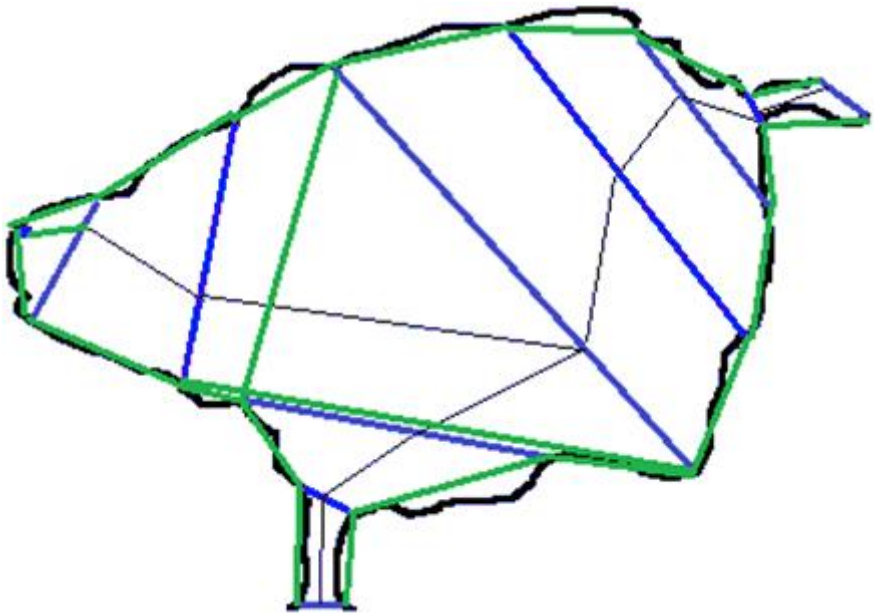


Иллюстрация 12: Съемка залов разумных размеров производится так же как съемка ходов, гроты можно снимать добавляя развилку и тупик. Метод съемки чрезмерно больших залов приведен в разделе 4.1.

4. Алгоритм работы программы

4.1. Триангуляция

Введем термин «отстрел пикета» - вектор от пикета до некоторой точки стены. У каждого пикета есть свой набор отстрелов (быть может пустой), у каждого отстрела есть свой пикет. Под положением пикета будет подразумеваться его положение в пространстве относительно некоторого выбранного начала координат.

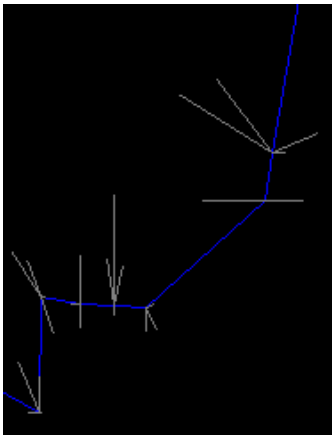


Иллюстрация 13: Пикеты с отстрелами

Задача построения объемного представления топосъемки сводится к задаче построения объемного представления на участках между всеми парами соседних пикетов, что сводится к задаче соединения отстрелов каждого пикета с отстрелами соседнего для него пикета полигонами. Использовать будем треугольные полигоны, поэтому в нахождении троек отстрелов, задающих эти треугольники, и заключается задача, которую мы будем называть задачей триангуляции.

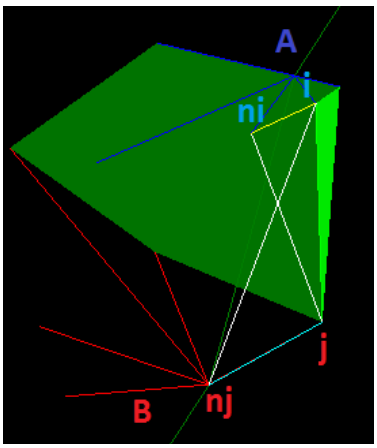


Иллюстрация 14: Решение задачи триангуляции для пары соседних пикетов

Задачу триангуляции для двух пикетов A и B будем решать следующим образом:

1) зададим правило выбора начальных отстрелов bi, bj.

2) зададим порядок обхода отстрелов одного пикета A и некоторый порядок обхода другого пикета B

Если дана некая пара отстрелов i и j пикетов A и B соответственно, называемая текущей парой, то существуют отстрелы ni, nj, идущие следом за отстрелами i и j в соответствии с заданным в п.1 порядком, и должно быть

3) задано некоторое правило выбора между двумя треугольниками (i, j, ni), (j, i, nj)

Тогда, двигаясь по отстрелам пикетов A и B в соответствии с порядком 2, начиная с отстрелов выбранных правилом 1, и принимая для перехода на следующую итерацию значения i = ni или j = nj в зависимости от выбранного алгоритмом 3 треугольника, мы получим набор «выбранных» треугольников, являющихся решением задачи триангуляции для пикетов A и B. Соответственно, надо найти правило нахождения порядка 2, а также правила выбора 1 и 3.

4.1.1. Выбор порядка обхода отстрелов

Самый простой порядок обхода — по часовой стрелке, но для этого нужна ось вращения. Вспомним, что все отстрелы делались из точки местоположения пикета, а также условие, что отстрелы представляют характерное сечение хода в окрестности пикета. Поэтому в качестве оси вращения мы возьмем вектор нормали к воображаемой плоскости, образующей сечение хода, проведенный через местоположение пикета.

В качестве вектора нормали возьмем среднее перпендикуляров (векторных произведений) ко всем парам векторов пикет-отстрел. В качестве точки отсчета — вектора 12 часов можно взять любой ненулевой, даже случайный, вектор в плоскости сечения — назовем его az для пиктеа A и bz для пикета B. Отсортировав отстрелы по возрастанию угла между вектором пикет-отстрел и вектором 12 часов получим искомый порядок обхода.

4.1.2. Выбор пикетов для начала итерационного процесса и правила сравнения двух полигонов

Возьмем среднее между двумя нормалями, определяющими плоскости сечения хода для пикетов A и B из главы 4.1. - получим третью плоскость C. Спроецируем на эту плоскость отстрелы пикетов A и B и, имея порядок обхода для отстрелов, получим два двухмерных многогранника D и E. Оба многогранника в общем случае являются невыпуклыми, что мешает дальнейшим рассуждениям, поэтому постоим еще два многогранник cD и cE на основе D и E отбросив все из вогнутые вершины. Отброшенные вершины и то, откуда они были выброшены - запомним — они еще пригодятся.

Дальше в угоду простоте рассуждений допускается много математических неточностей, опускается много граничных эффектов.

Зададим теперь произвольное направление на плоскости C единичным вектором z.

Введем функцию angleTo(vec1, vec2) возвращающую угол, на который нужно повернуть vec1, чтобы он совпал с vec2

Введем, что cD[i] – положение вершины i многоугольника cD

Вектором вершины многогранника vcDi назовем вектор от предыдущей вершины этого многогранника на эту, т.е, vcDi = cD[i] – cD[i-1]

Выберем в качестве нулевой вершины многогранников cD[i] вершину с минимальным значением angleTo(z, vcDi). Т.е. вершину для которой ее вектор находится с одной стороны вектора z, а вектор предыдущей вершины с другой стороны z.

Это сделано для того, чтобы отметить один важный факт — функция angleTo(z, vcDi) является монотонно возрастающей при i принадлежащей [0..size(cD)). Это условие выполняется для любого выпуклого многоугольника, но не выполняется для невыпуклого.

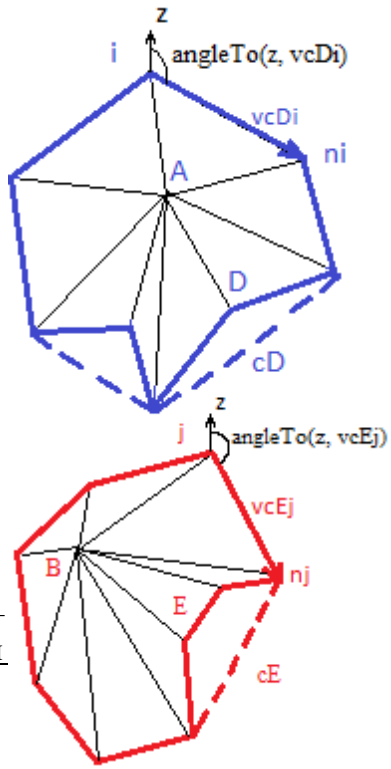


Иллюстрация 15: Многоугольники D, cD, E, cE - проекции отстрелов пикетов A и B на плоскость C в процессе решения задачи триангуляции

Говоря простым языком ребра выпуклого многоугольника всегда все более заворачивают в одну сторону — в нашем случае по часовой. Именно по тому насколько заворачивает ребро многоугольника cD вершины i по отношению к тому на сколько сильно заворачивает ребро многоугольника cE вершины j мы и будем сравнивать их. Т.е. будем сравнивать значения angleTo(z, vcDi) и angleTo(z, vcEj).

Рассмотрим алгоритм.

Мы перечисляем вершины i и j в порядке их вхождения в многоугольники cD и cE. В начале каждого шага мы знаем что вершины i и j уже вошли в крайний полигон и нам нужно выбрать между тем, добавить ли полигон (i, j, ni) и увеличить на единицу i для следующего шага, или добавить полигон (j, i, nj) и увеличить на единицу j для следующего шага, т.е. нам нужно выбрать по какому многоугольнику продвинуться дальше на одну вершину. Идея состоит в том, что если многоугольник cD на данном участке заворачивает больше многоугольника cE, то нам еще рано двигаться дальше по многоугольнику cD, а в место этого стоит обработать вершину многоугольника cE. Таким образом, сравнивая значения функций angleTo(z, vcDni) и angleTo(z, vcEnj), мы принимаем решения до тех пор, пока не придем обратно к начальным отстрелам.

Перейдем теперь к выбору начальных отстрелов. Нам нужно выбрать, отстрелы с

какими номерами i и j являются стартовыми для нашего алгоритма. Зная принцип выбора между двумя следующими значениями, легко придумать его и для начальных. Начальные значения i и j возьмем так, чтобы угол между векторами vcDi и vcDj был минимален, или , другими словами, возьмем вершины двух "максимально параллельных" ребер.

Пора вспомнить, что для того, чтобы работать с выпуклыми многоугольниками cD и cE, мы исключили "вогнутые" вершины из многоугольников C D.

Сейчас уже должно быть понятно, что если бы мы это не сделали — наш алгоритм на каком-нибудь из шагов попал бы в «потенциальную яму» для одной из "вогнутых" вершин и оставался бы там до тех пор, пока не завернул бы достаточно сильно по второму многоугольнику, что привело бы к неправдоподобному «скручиванию» топосъемки.

Для того чтобы включить опущенные вершины в нашу триангуляцию необходимо просто заменить всего полигоны (i, j, ni), где ni не равно i+1 на серию полигонов (i, j, i+1), (i+1, j, i+2)....(ni-1, j, ni).

4.2. Сглаживание

Для работы режима отображения объемов без сглаживания служит описанный в 5.1 алгоритм. Для того чтобы получить гладкую, бесшовную картинку, показываемую в режиме сглаженного отображения объемов, необходимо ввести дополнительный шаг обработки полигонов.

Естественно, что каждый отстрел входит как вершина во множество полигонов. У каждого полигона своя нормаль. Беря среднее этих нормалей мы получим нормаль для отстрела.

Имея полигон и нормали для каждой его вершины, мы можем построить поверхность Безье. Процесс подробно описан в этих статьях:

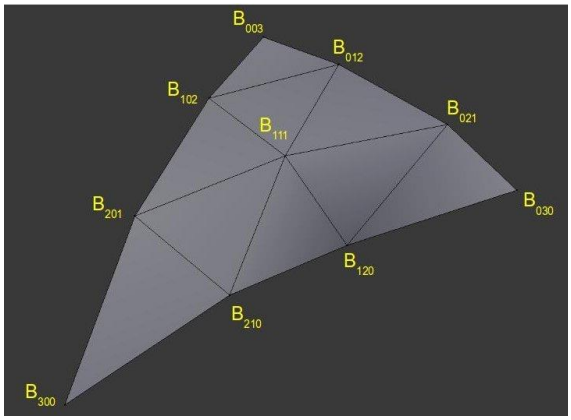


Иллюстрация 16: Расположение контрольных точек при построении поверхности безье

- http://triplepointfive.github.io/ogltutor/tutorials/tutorial31.html
- http://onrendering.blogspot.ru/2011/12/tessellation-on-gpu-curved-pn-triangles.html

Далее аппроксимируем поверхность Безье набором треугольником — полигонов. Отображаем в программе полученные полигоны вместо полигонов полученных на этапе 4.1. Получаем гладкую, почти бесшовную поверхность.

Регулируя положение контрольных точек для поверхности Безье и количество полигонов, используемых для построения поверхности, мы можем задавать степень сглаживания.

4.3. Интерполяция данных положения стен для пикетов без соответствующих данных на основе соседних пикетов

Реализовано 3 режима интерполяции.

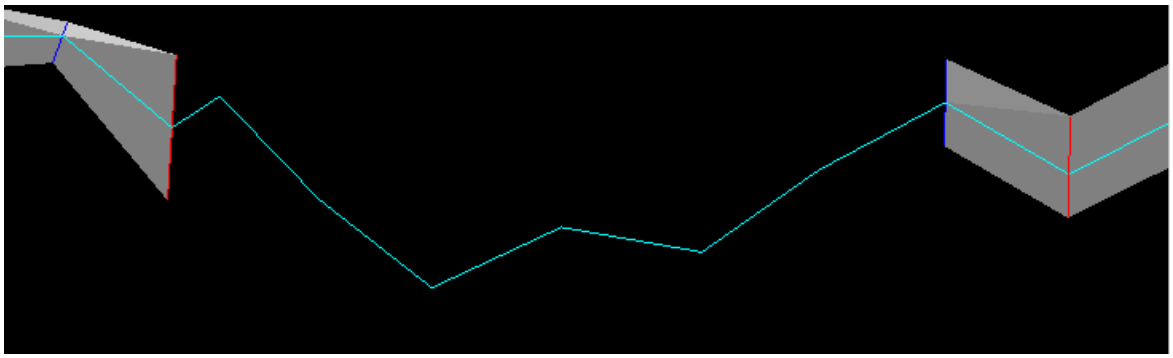


Иллюстрация 17: Участок съемки с отсутствующими данными о положении стен

Первый режим по сути никакой интерполяцией не является; его работа состоит в том, что мы пикеты, между которыми есть пикеты без отстрелов, считаем соседними и соединяем их отстрелы полигонами игнорируя существование промежуточных пикетов.

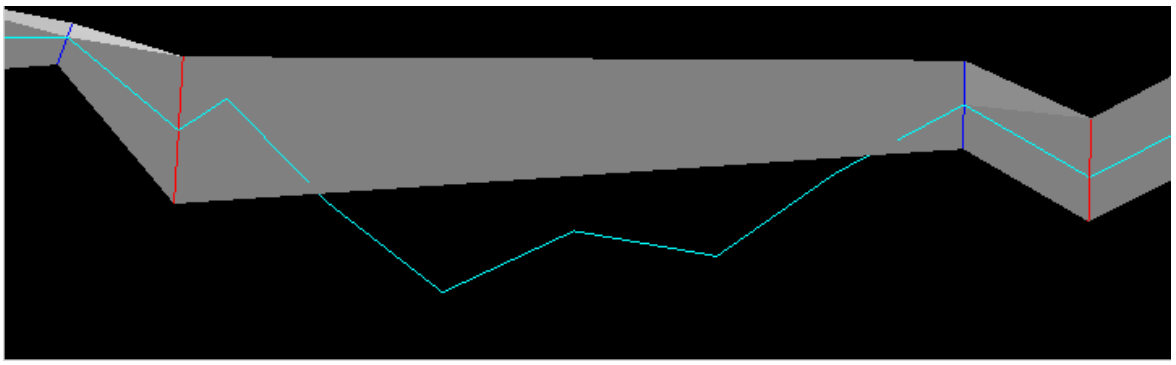


Иллюстрация 18: Режим игнорирования промежуточных пикетов

Второй режим - режим интерполяции с простыми смещением заключается в том, что мы соединяем полигонами отстрелы пикетов, имеющих отстрелы, как будто они соседние, но не отправляем эти полигоны на отрисовку. Находим для каждого пикета без отстрелов пересечение плоскости, перпендикулярной направлению хода на этом пикете с ребрами построенных полигонов. Смещаем все точки пересечения так, чтобы пикет оказался в их центре. Смещенные точки пересечения и будут отстрелами пикета.

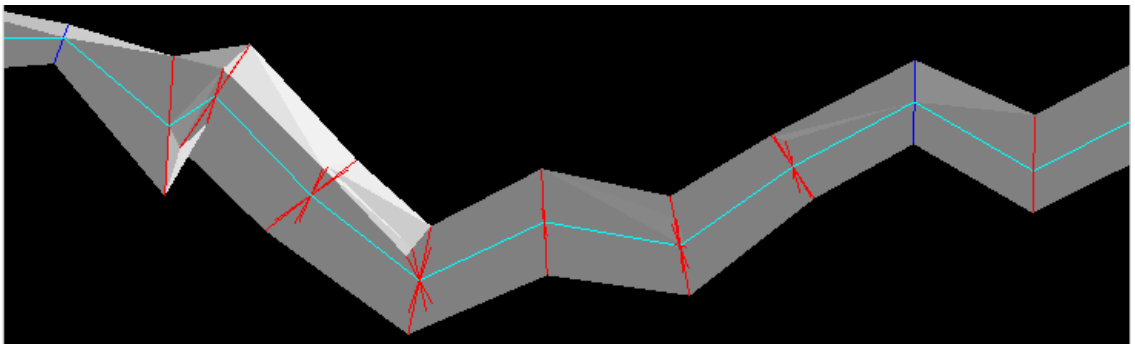


Иллюстрация 19: Режим интерполяции с простым смещением

Для третьего режима выполняем все то же что и для второго, но смещаем не до тех пор, пока пикет окажется в центре, а до тех пор пока пикет не окажется внутри

многоугольника образованного новыми отстрелами. Логика размышлений состоит в том, что мы не знаем в каком месте снимаемого хода находится пикет. Единственное что мы знаем что он не может быть снаружи, поэтому стремимся получить ход наиболее простой конфигурации для которого выполняется данное требование — все пикеты находятся внутри.

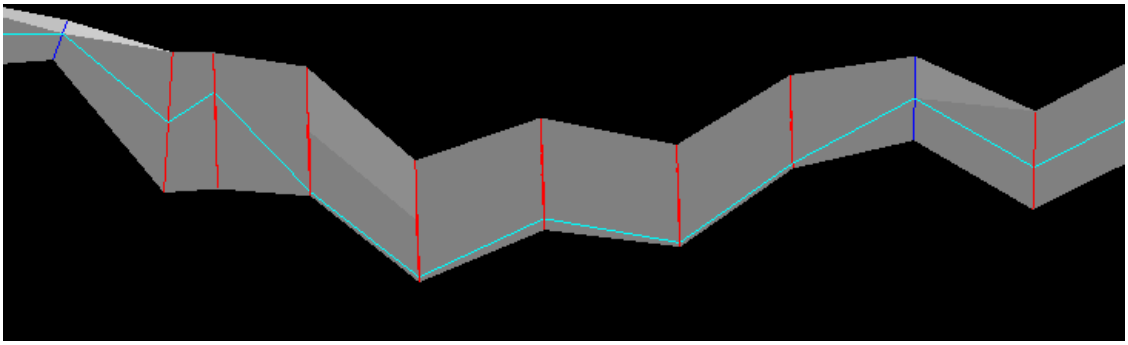


Иллюстрация 20: Режим интерполяции со смещением до границы