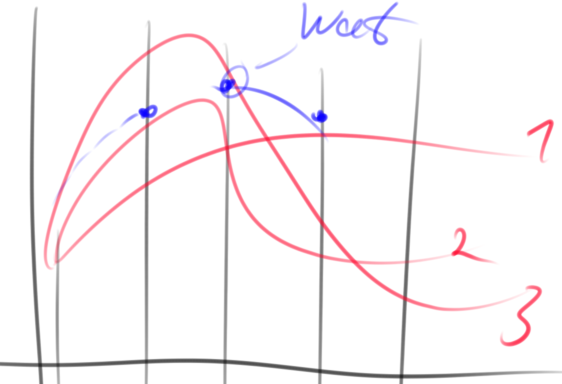
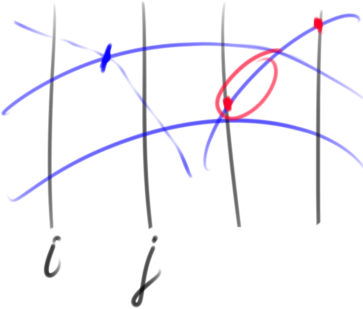
21.05.14-----------------------------------------------------------------------

Вычисления могут производиться не с шагом=1 по оХ. Если очередная точка является невидимой, то она не соединяется с предыдущей. Наоборот, если видима, то соединяется. Для упрощения вычислений можно использовать линейную аппроксимации.





Требуется искать точки пересечения кривых. Для облегчения вычислений, можно использовать линейную аппроксимацию кривых.

Y\_тек = Y\_тек(xi) + m\_тек(x-xi)

Y\_пред = Y\_пред(xi) + m\_пред(x-xi)

M\_тек = (Y\_max(xj) – Y\_тек(xi)) / (xj – xi)

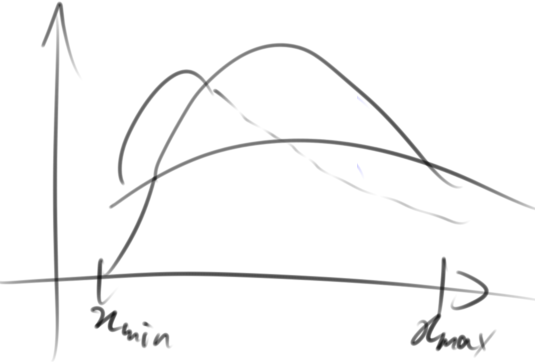
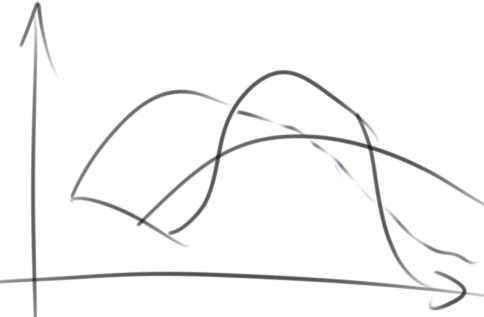
M\_пред = (Y\_пред(xj) – Y\_пред(xi)) / (xj – xi)

Y\_тек(xi) + m\_тек(x-xi) = Y\_пред(xi) + m\_пред(x-xi)

M – тангенс угла наклона одной прямой и другой. Текущее – для рассматриваемой кривой; предыдущая кривая - либо верхний либо нижний горизонт, в зависимости от того где расположена текущая.

Определяем х из упред и утек:

; , dx=xj-xi

После поворота будет ->

1.Если точка P является первой на текущей кривой, то:

- если эта кривая первая, то запомнить текущую точку Q=P

- если кривая не первая, то соединить точку P с точкой Q, заменить Q=P

Последовательность действий при реализации алгоритма плавающего горизонта:

1. обработать левое боковое ребро (^)

2. для каждой точки каждой кривой выполнить:

2.1 если значение Y(xi) для этой точки больше Y\_max(xi), т.е. больше

значения в массиве верхнего горизонта, или меньше Y\_min(xi), т.е. меньше

значения в массиве нижнего горизонта

текущая точка кривой является видимой

иначе

текущая точка является невидимой

2.2 если видимость кривой на очередном интервале изменяется

найти точку пересечения текущей кривой с линией верхнего\нижнего горизонта

2.3 если на очередном интервале кривая видима

отрисовывается весь участок

если текущая точка видима, а предыдущая невидима

отрисовывается участок от пересечения до текущей

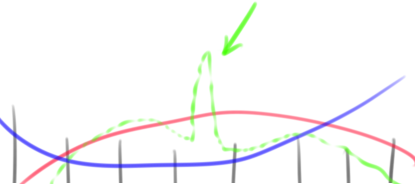
если текущая невидима, а предудущая видима

отрисовывается участок от предыдуще до пересечения

3. изменить содержимое текущих элементов верхнего и нижнего горизонтов (если текущая больше верхнего или меньше нижнего)

4. обработать правое боковое ребро

Выбор шага дискретизации зависит от характеристик функции (кривой) – второй производной. При большом шаге можно потерять кусок поверхности.



В целом, данный алгоритм не справляется с общим случаем задачи об отсечении невидимых частей.

**Алгоритм Z-буфера**

Помимо буфера кадра, используется буфер, хранящий информацию о координате Z каждого пикселя (ближайшго на текущий момент времени объекта, связанного с этим пикселем). С высокой точностью хранить Й не обязательно – достаточно лишь различать поверхности.

Идейно алгоритм прост – не требуется никаких сортировок (выигрыш по времени), а с позиции удобства программисту – объекты сцены можно обрабатывать в произвольном порядке. Тем не менее, в Й-буфере не учитывается эффект прозрачности или сглаживания.

1. проинициализировать буфер кадра фоновым значением

2. проинициализировать й-буфер минимальным значением глубины

3. преобразовать в растровую форму каждый многоугольник сцены в произвольном порядке - определение принадлежности пикселей экрана внутренней области многоугла.

Для каждой точки многоугольника (ХУ) вычислить её глубину (Й)

сравнить значения Й\_буф(ху) и Й(ху). если Й(ХУ) > Й\_буф(ху), то

занести в Й-буфер текущее значение Й(ху); занести в буфер кадра в соответствующую позицию цвет текущего многоугольника

4. если (Й(ху) > Й\_буф(ху)) И (Й(ху) <= Й\_сек.плоск) то можно построить разрез.

Помимо отсутствия сортировок, алгоритм Й-буфера позволяет построить и разрез объекта.

Уравнение плоскости: Ax+By+Cz+D=0. Для прямоугольника , С!=0. Если С==0, то прямоугольник расположен параллельно вектору взгляда – отрисовывать нужно линию или вершину. Используется уравнение ребра; , если координаты не равны. Если же равны, ребро горизонтально, отрисовывается точка.

.

**Алгоритм со списком приоритетов (алгоритм художника)**

Основная идея – объекты отрисовываются «издали – вблизь», начиная с наиболее удалённых от наблюдателя, до ближайших. Основная задача: перед изображением объекта определить, может ли очередной прямоугольник загораживать прямоугольники, отображаемые в дальнейшем. Если не может загораживать другие, то он отрисовывается; если может, то проводятся расчёты.