1. Зачем эти строки

bool t0i = (t0 & mask) != 0;

bool t1i = (t1 & mask) != 0;

Коды уже вычислены до этого.

Это обращение к i-ому биту 4хбитного кода вершины.

Mask изначальна равна 1000  
  
на после каждой итерации mask делиться на два, тем самым mask будет иметь вид  
1000 на 0ой итерации  
0100 на 1ой  
0010 на 2ой  
0001 на 3ей

Если применить побитовое И маски с вычисленным кодом вершины получиться либо маска (если разряд равен единице), либо ноль (разряд равен нулю)

1. Эти две проверки лучше разделить       if (!isVert && i < 2). Если вертикальный, то сразу ясно, что делать.

Объясните это утверждение.

Если отрезок вертикальный, то точки с пересечения с окном находятся тривиально, координата x вершины и координата y границы окна.

1. Почему, если очередные разряды кодов равны, то уходим на следующий шаг цикла?

Возможны три варианта конфигурации разрядов кодов. Вариант, когда оба равны единицы, невозможен (иначе отрезок шагом выше был бы распознан как невидимый и алгоритм закончил свою работу).

Если очередные разряды кодов равны нулю, то обе вершины расположены по видимую сторону отсекателя, следовательно, пересечение с текущей границей отсекателя искать не надо.

Мы должны обрабатывать только случаи, когда вершина расположена по видимую сторону, другая по невидимую.

1. Как решается вопрос о том, какую часть отрезка надо отбросить после нахождения пересечения?

Надо отбросить ту часть отрезка, которая расположена по невидимую сторону границы. Так как точка P1 находится вне границы, то достаточно переместить точку P1 в найденную точку пересечения.

1. Чем этот алгоритм отличается от простого?

При нахождении точки пересечения отрезка со стороной отсекателя не производиться анализа найденной точки пересечения на корректность, в простом алгоритме после того как найдена точка пересечения отрезка с очередной границей отсекателя проверяется попала ли найденная точка в пределы отсекателя.