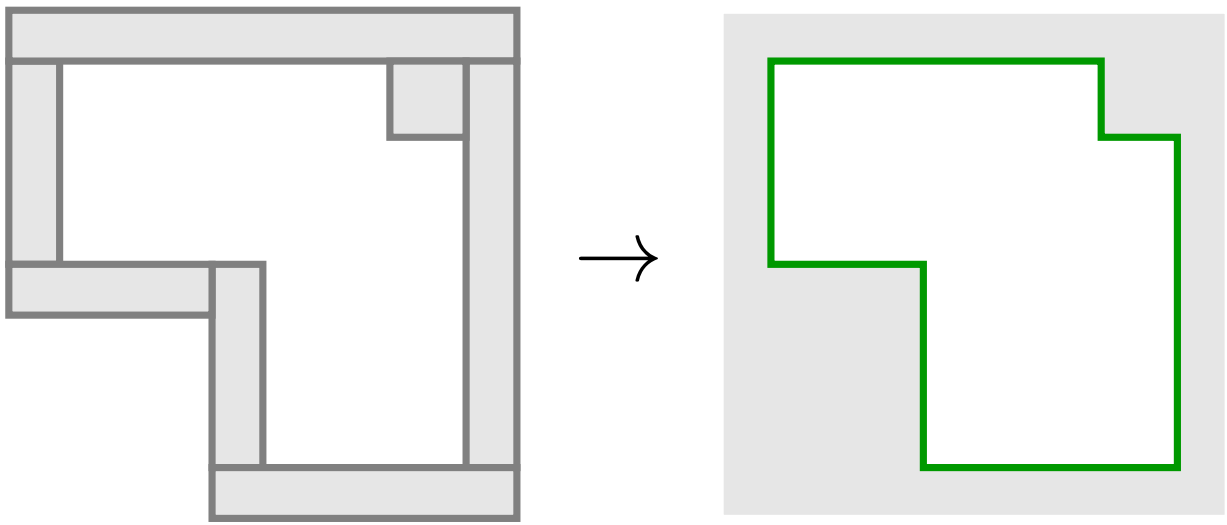
# CSG-геометрия

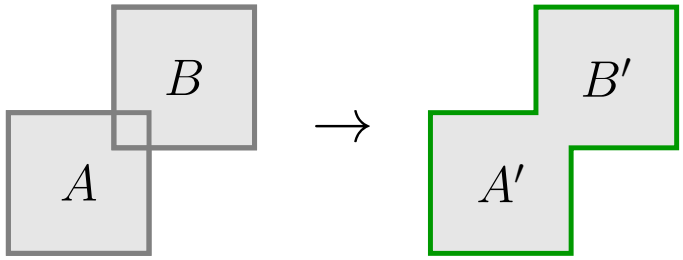
- Браш это выпуклый многогранник. Геометрия мира формируется из брашей вручную:



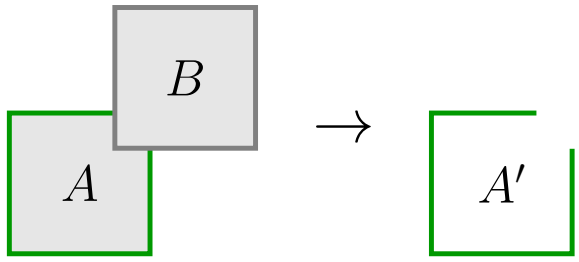
- Карта, составленная из брашей, имеет слишком много полигонов. Задача CSG-алгоритма – перевести множество брашей в замкнутую поверхность из полигонов. Игровой мир находится внутри этой поверхности, за ней находится заполненное пространство.

# CSG-геометрия: объединение двух брашей

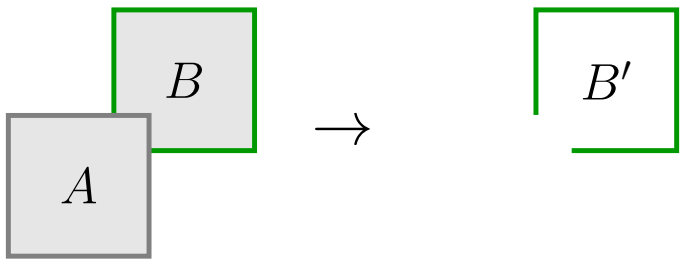
- Пусть нужно объединить браши $A$ и $B$:




Сопоставим $A$ с $B$, получив набор полигонов $A'$:




Сопоставим $B$ с $A$, получив набор полигонов $B'$:




Объединяем $A'$ и $B'$, получаем результат.

# CSG-геометрия: объединение брашей уровня

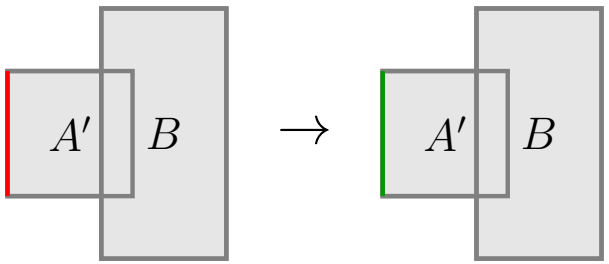
- Для компиляции уровня из списка брашей необходимо сопоставить каждый браш с каждым:

**def** `csgCompile(`$brushes$`):`
  $brushesCopy \leftarrow \mathrm{copy}(brushes)$;
  **for** $A' \in brush$**:**
    **for** $B \in brushesCopy \setminus A$**:**
      $A' \leftarrow \mathrm{clip}(A', B)$;

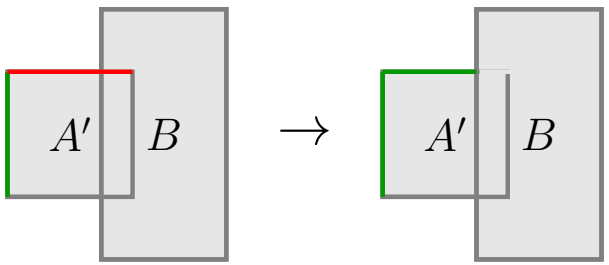
- В итоге $brushes$ будет хранить итоговый набор полигонов уровня.

# CSG-геометрия: разрез браша брашем

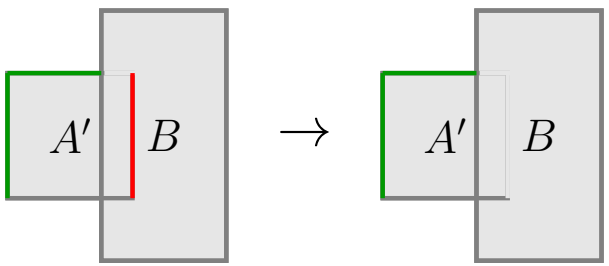
- В функции $\mathrm{clip}$ из каждого полигона $b1$ необходимо вырезать ту часть что находится внутри $b2$. Переберём полигоны $A'$: берём левый полигон $A'$ – он находится вне $B$ и не пересекается с ним – игнорировать:




Берём верхний полигон $A'$, он пересекается полигонами из $B$ – разрезать и отбросить правую часть:

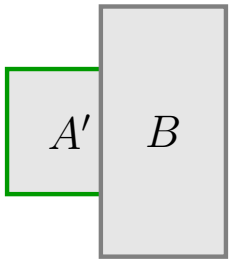



Берём правый полигон $A'$, он полностью внутри $B$ – удалить:

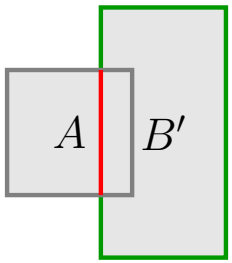


# CSG-геометрия: разрез браша брашем

- В итоге получаем $A'$:




- В случае сечения $B'$ при помощи $A$ все стороны сохранятся кроме левой: она будет разрезана на три части, центральная из которых удалится поскольку находится внутри $A$:



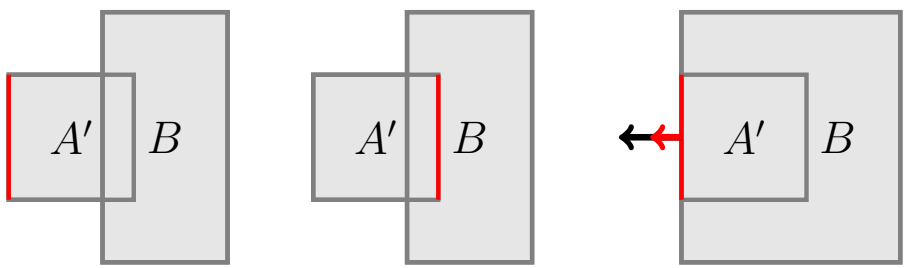

- Псевдокод clip выглядит так:

**def** `csgCompile(`$A'$, $B$`):`
  $newPolygonList \leftarrow []$;
  **for** $p \in A'.polygons$**:**
    clipPolygon$(A'.polygons, p, B)$
  **return** $newPolygonList$

# CSG-геометрия: разрез полигона брашем

▶ Функция разреза полигона $p$ набором полигонов $B$ работает так: определяется положение $p$ относительно $B$:

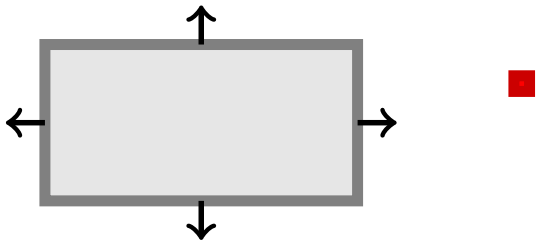
1. Если $p$ находится снаружи $B$, то $p$ возвращается в не изменённом виде (рис 1.);
2. Если $p$ находится внутри $B$, то ничего не возвращать (рис 2.);
3. Если $p$ лежит на $q$, одном из полигонов $B$, и их нормали совпадают, вернуть один из них чтобы не образовалось дыры, а другой полигон будет рассечён позже (рис 3.). Обычно возвращают полигон того браша, который сечётся позже;
4. Если три предыдущих проверки не прошли, $p$ точно пересекается одним или несколькими полигонами $L \subseteq B.polygons$ браша $B$.



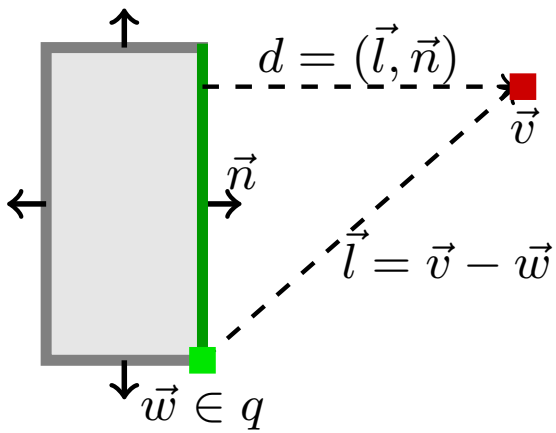

1. 2. 3.

# CSG-геометрия: относительное положения точки

- В силу выпуклости браша возможна простая проверка положения точки относительно браша: если точка находится перед любым полигоном браша, то она находится снаружи.



- Для каждого полигона $q \in B.polygons$ браша $B$ возьмём произвольную точку $\vec{w} \in q$. Тогда расстояние $d$ вдоль нормали $q$ от точки $\vec{v}$ до плоскости полигона $q$ будет равно скалярному произведению $d = (\vec{n},\ \vec{v} - \vec{w})$.




Число $d$ будет больше нуля если $w$ находится перед полигоном и меньше нуля если за ним. Точка $\vec{v}$ будет внутри браша если $d < 0$ для всех полигонов браша.

# CSG-геометрия: относительное положения полигона

- Та же логика применима и к определению положения полигона $p$.

Переберём полигоны $q \in B.polygons$:

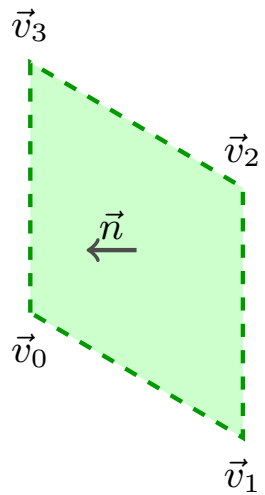
1. Если для всех вершин $v \in p$ расстояние $d = 0$ то полигон лежит на плоскости;
2. Если для всех вершин $d \geq 0$, то полигон $p$ точно снаружи, его нужно вернуть;
3. Если для всех вершин $d \leq 0$, то полигон $p$, возможно, внутри. Если на данный момент перебраны уже все $q$, то $p$ находится внутри $B$ и его следует удалить (вернуть ничего);
4. В ином случае имеет место быть пересечение $p$ и $q$.

Полигон $p$ может пересекаться с несколькими полигонами $S \subset B.polygons$. В программе полигон последовательно режется каждым из них:

1. $p$ режется $q \in S$ на две половинки $p_1, p_2$;
2. Из двух половинок в силу выпуклости $B$ половинка $p_1$ лежит вне $B$, а $p_2$ пересекается $B$ или находится внутри $B$, а значит в последующих итерациях рассматривается только $p_2$.

# CSG-геометрия: полигон

- Вершины полигона хранятся так, что любые две последовательные вершины образуют правую тройку с нормалью плоскости:
$\vec{n} = \langle \vec{v}_1 - \vec{v}_0,\ \vec{v}_2 - \vec{v}_0 \rangle$.




- UV-координаты так-же рассчитываются на основе этого порядка вершин.

# CSG-геометрия: полигон

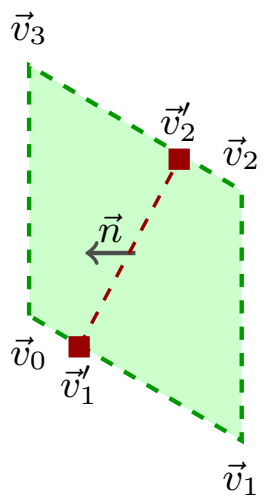
- При разрезе полигона $p$ плоскостью $q$ мы проходим каждое ребро (последовательные пары вершин) $v_{i+1} - v_i$. Если $d_1$ это расстояние от плоскости до $v_i$, $d_2$ – до $v_{i+1}$, то точка пересечения может быть найдена как

$$v' = \vec{v}_i + (\vec{v}_{i+1} - \vec{v}_i)\frac{d_1}{d_1 + d_2}, \quad d_1 = (\vec{n}, \vec{v}_i - w),\ d_2 = (\vec{n}, \vec{v}_{i+1} - w),\ w \in q.$$

Точка $v'$ будет расположена на ребре если $d = \frac{d_1}{d_1+d_2} \in [0, 1)$.

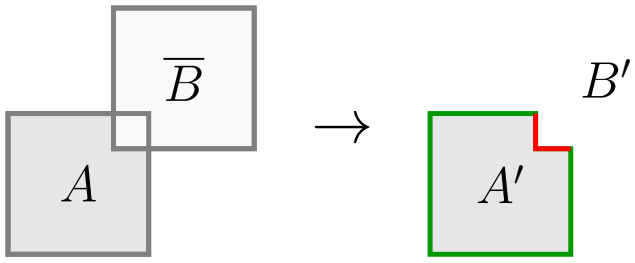
- После разреза останется две вершины: $\vec{v}'_1$ и $\vec{v}'_2$. Поскольку рёбра, на которых расположены эти точки, известны, можно:

1. Сохранить нормаль обоих полигонов путём сохранения порядка вершин;
2. Мгновенно определить какая половинка находится за секущей плоскостью а какая перед нею.

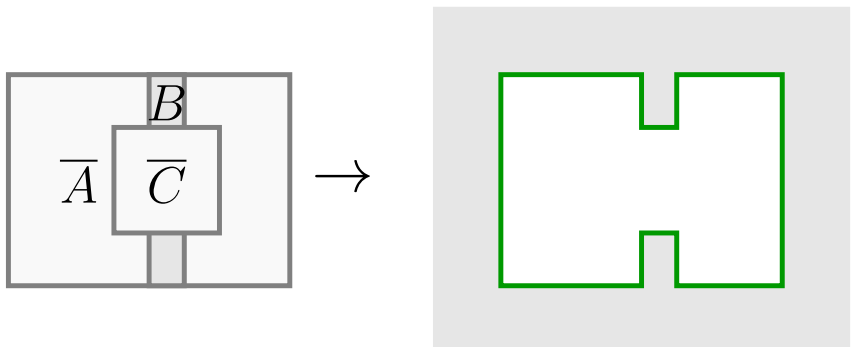


# CSG-геометрия: комнаты

- Помимо заполняющих (твёрдых) брашей есть и браши, вырезающие пространство (полые):



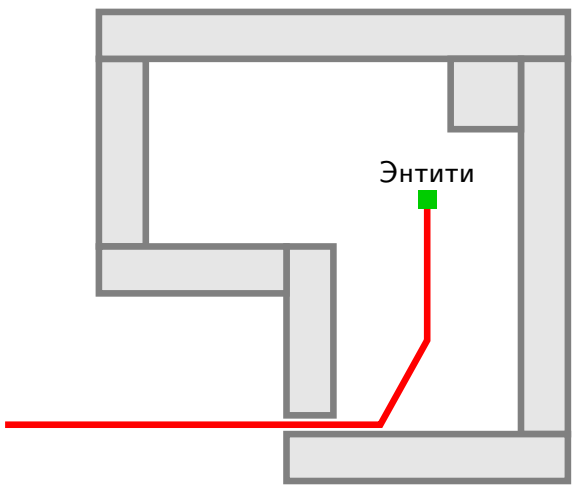

Полые браши вырезают пространство, затрагивая только предыдущие браши.




- В движках мир изначально может быть пуст или заполнен. В заполненном пространстве для создания комнаты необходимо её вырезать полым брашем, в пустом нужно построить 6 стен из твёрдых брашей или поставить большой твёрдый браш и вырезать комнату внутри него.

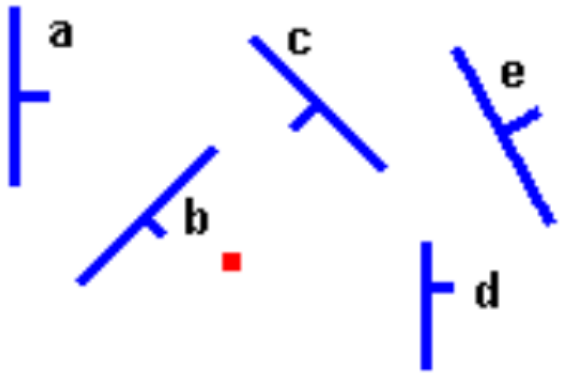
# BSP дерево

- BSP дерево представляет собой рекурсивное иерархическое разбиение пространства на выпуклые области. Оно строится заранее и не подлежит изменению. Оно позволяет:
  1. Сортировать полигоны по удалённости к камере;
  2. Отсеивать невидимые из точки области карты;
  3. Контроль столкновений.
- BSP-компилятор переводит замкнутую поверхность из полигонов в бинарное дерево. Итоговая геометрия уровня обязана быть замкнутой, не должно быть дыр во внешнюю пустоту. Дыры в пустоту называют ликами (leak):



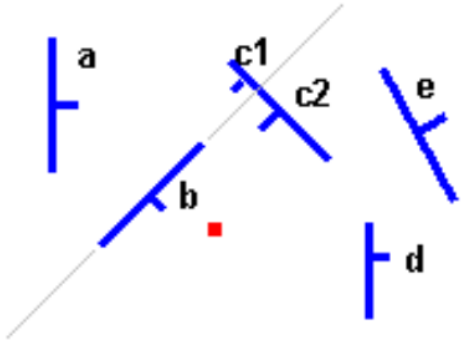
# BSP дерево: построение

- Пусть есть набор полигонов вида




- Схема алгоритма:
  1. Выбор *сплиттера* среди входных полигонов;
  2. Рассечение набора полигонов на пространство перед сплиттером и за ни;
  3. Вызов алгоритма для каждой из половинок, продолжать пока входной набор полигонов не будет состоять из одного полигона;

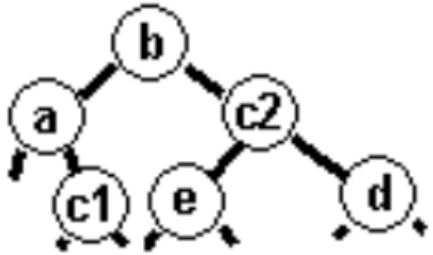
# BSP дерево: построение



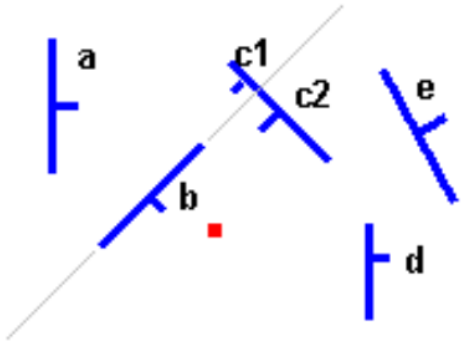

▶ Выбрав сплиттер, мы можем при помощи алгоритма, использованного при построении CSG-геометрии, поделить остальные полигоны на два множества:

1. Те что стоят со стороны нормали сплиттера или коплонарны ему и совпадают с ним нормалью переходят в первое множество;
2. Те что стоят сзади сплиттера или коплонарны ему и противоположны с ним по нормали переходят во второе множество;
3. Если полигон пересекается плоскостью сплиттера, то он рассекается.

Полигон сплиттера кладётся в вершину.



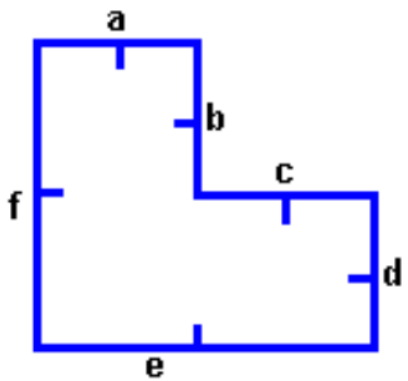
# BSP дерево: построение

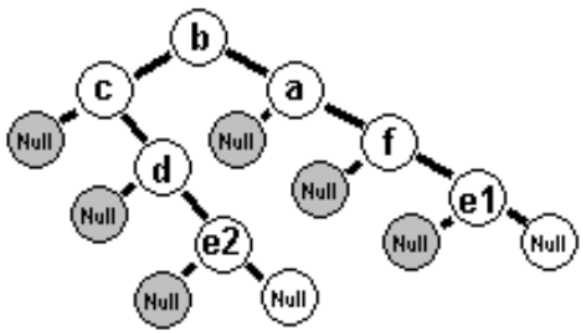



- Выбор сплиттера не случаен. От стратегии выбора зависит размер дерева. Полный перебор вариантов NP-сложен, потому применяют метрические оценки, например стремятся уменьшить разницу между число полигонов перед полигоном и за ним:

$$\min |front - back|.$$

- Пример:



# BSP дерево: отрисовка

- BSP дерево позволяет сортировать полигоны в порядке удалённости.