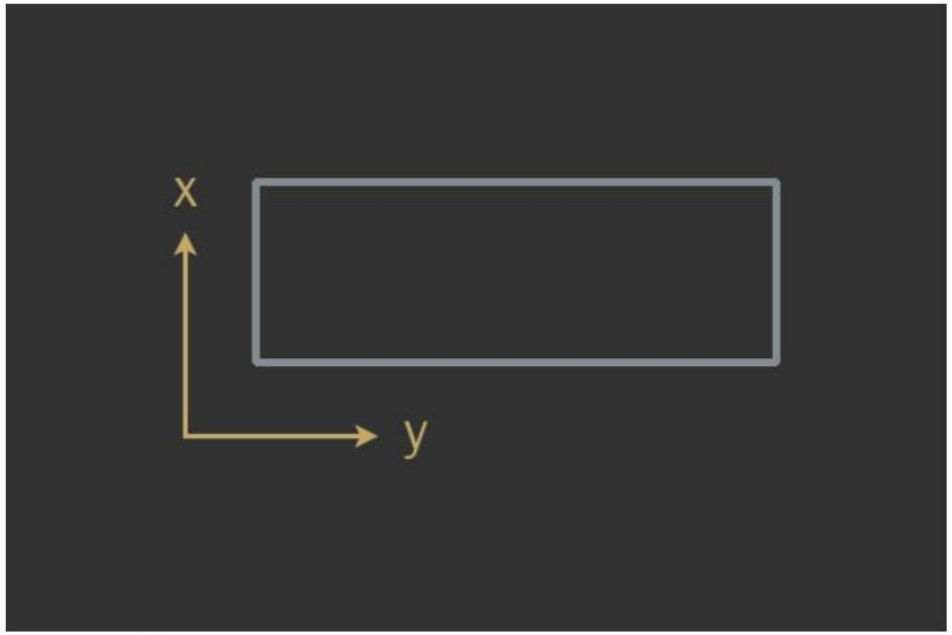
# Поиск коллизий

# 1)Ограничивающий прямоугольник, выровненный по координатным осям (AABB - Axis Aligned Bounding Box)

- Это значит, что прямоугольник не может вращаться и всегда находится под углом в 90 градусов. Обычно его называют «ограничивающим прямоугольником», потому что AABB используются для ограничения других, более сложных форм.



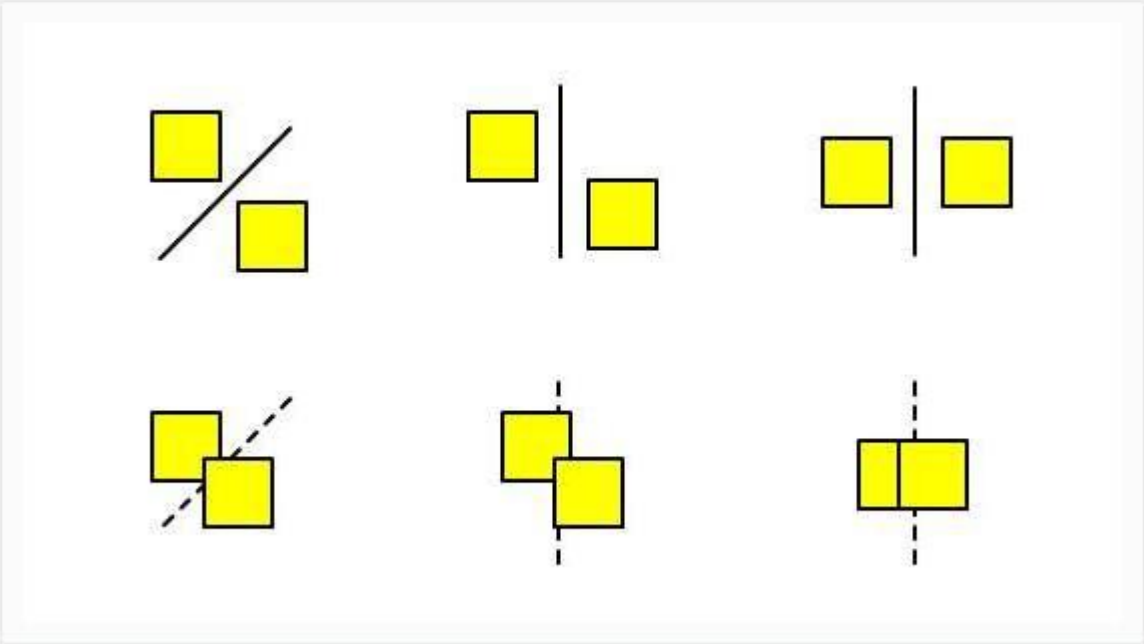

*Пример AABB.*

# Прямоугольник проще всего задать 2мя точками

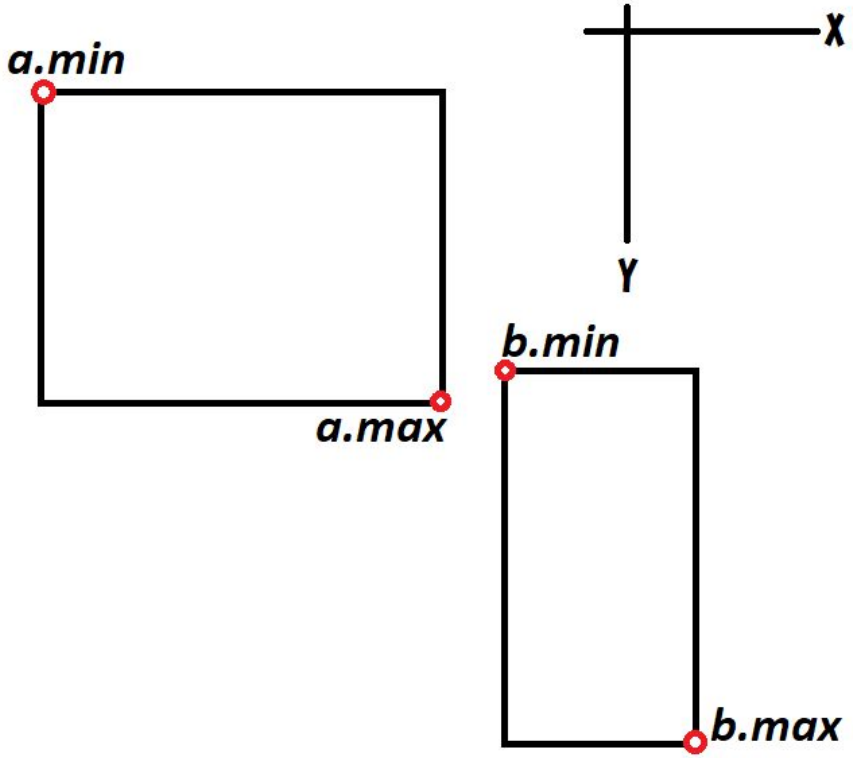
- Точка **min** обозначает нижние границы по осям x и y, а **max** обозначает верхние границы — иными словами, они обозначают верхний левый(0,0) и нижний правый углы.

```
struct AABB
{
  Vec2 min;
  Vec2 max;
};
```

Теорема, которая лежит в основе нахождения коллизий: **если есть линия, которая разделяет 2 объекта, то они не пересекаются**



# Простой алгоритм:




```
13 bool AABBvsAABB( AABB a, AABB b )
14 {
15   // Выходим без пересечения, потому что найдена разделяющая ось
16   if(a.max.x < b.min.x or a.min.x > b.max.x) return false
17   if(a.max.y < b.min.y or a.min.y > b.max.y) return false
18
19   // Разделяющая ось не найдена, поэтому существует по крайней мере одна пересекающая ось
20   return true
21 }
```

# 2) Окружность: задается координатой и радиусом

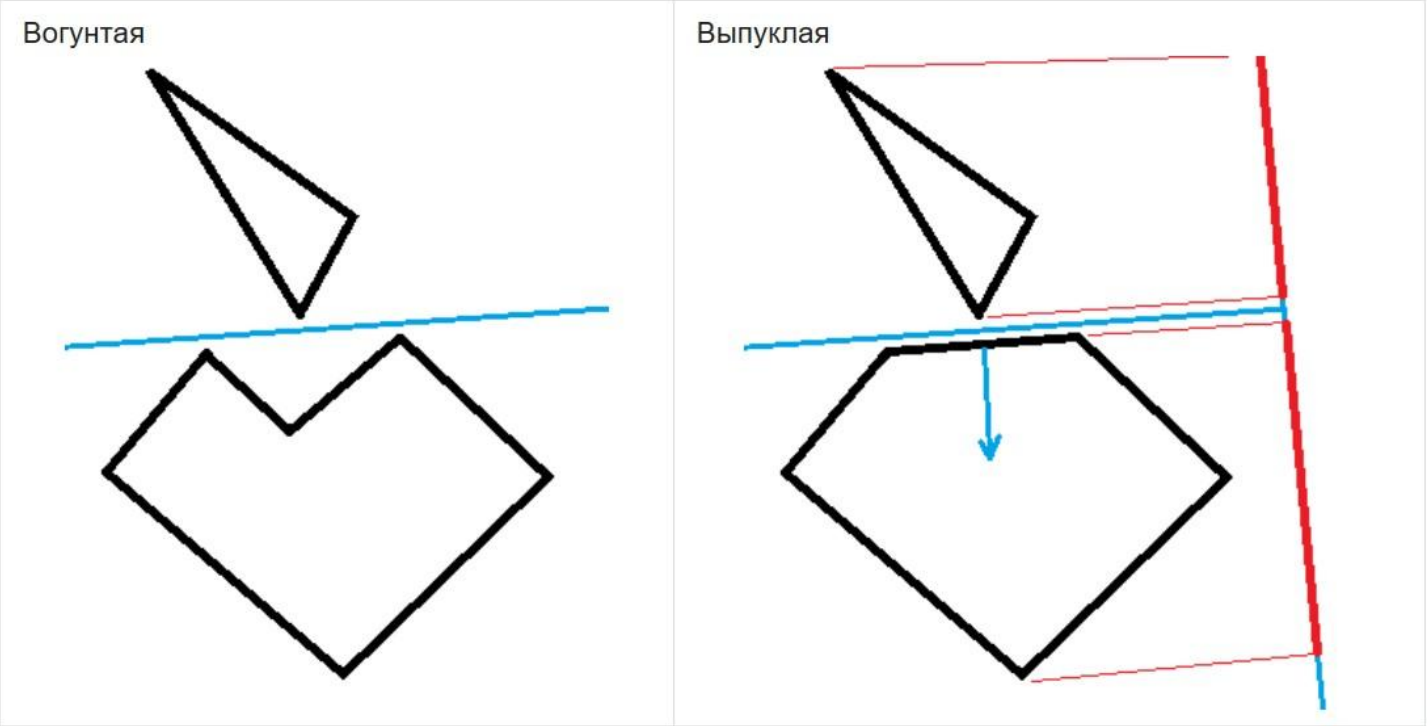
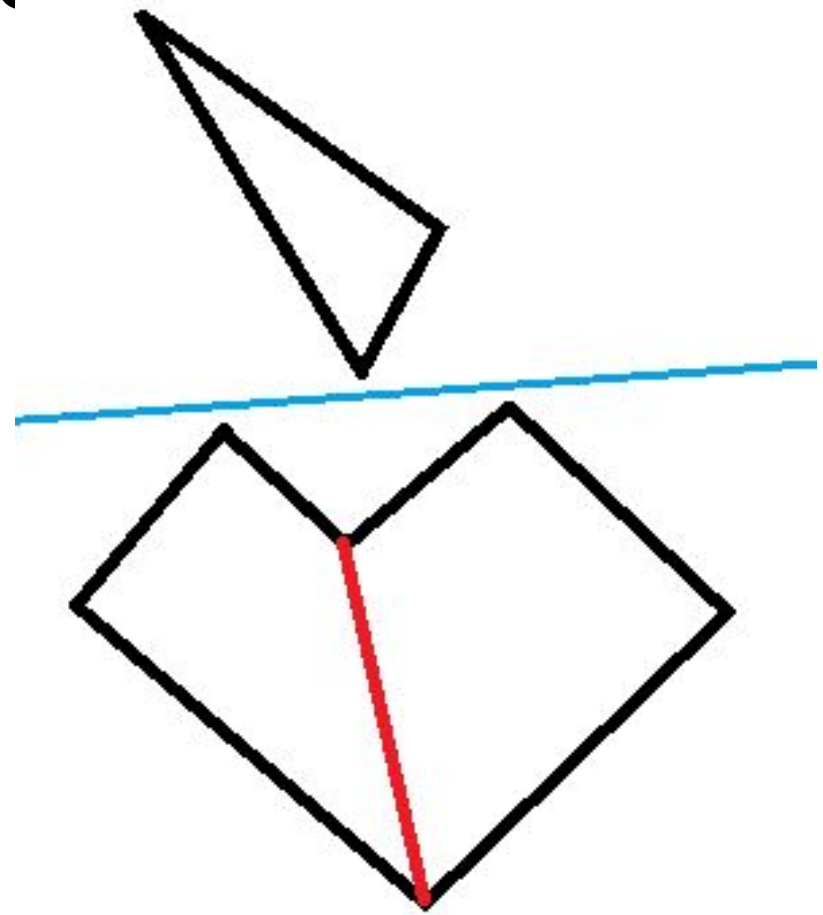
- Проверка пересечения двух окружностей очень проста: берём радиусы двух окружностей и складываем их, затем проверяем, больше ли эта сумма расстояния между двумя центрами окружностей.

  Важна только оптимизация, позволяющая избавиться от оператора квадратного корня

```
struct Circle
{
  float radius
  Vec position
};
```

```
13 float Distance( Vec2 a, Vec2 b )
14 {
15   return sqrt( (a.x - b.x)^2 + (a.y - b.y)^2 )
16 }
17
18 bool CirclevsCircleUnoptimized( Circle a, Circle b )
19 {
20   float r = a.radius + b.radius
21   return r < Distance( a.position, b.position )
22 }
23
24 bool CirclevsCircleOptimized( Circle a, Circle b )
25 {
26   float r = a.radius + b.radius
27   r *= r
28   return r < (a.x + b.x)^2 + (a.y + b.y)^2
29 }
30
```

# В общем случае(для не прямоугольников) идея такая же, нужно проверить можно ли провести линию(или плоскость для 3D) между объектами



Если фигура вогнутая, мы можем разделить ее на несколько выпуклых и проверить каждую их них по отдельности