Отрезки

Хранение отрезков

Тут ничего хитрого, храним их концы.

Пересечение отрезков

Возможные варианты взаимного расположения отрезков:

1. На одной прямой и
    1. пересекаются
    2. не пересекаются
    3. один содержит другой
2. Не на одной прямой и
    1. пересекаются
    2. не пересекаются

Чтобы понять, лежат ли отрезки на одной прямой, достаточно проверить, что концы одного из них лежат на прямой, задаваемой другим отрезком:

```
struct segment {
  array<pt, 2> ends;
};

// здесь прямую мы задаём 2 точками
bool pt::BelongsTo(line& L) {
  pt A = L.pts[0];
  pt B = L.pts[1];

  pt AB = pt(A, B);
  pt AP = pt(A, *this); // у вас должен быть написан конструктор pt (pt&, pt&)

  // векторное произведение
  return AB % AP == 0;
}

bool IsSegmentsOnSameLine(segment& s1, segment& s2) {
  line s1_line = line(s1.ends[0], s1.end[1]);
  for (int i = 0; i < 2; ++i) {
    pt s2_end = s2.ends[i];
    if (!s2_end.BelongsTo(s1_line)) {
      return false;
    }
  }

  return true;
}
```

Если отрезки лежат на одной прямой, то ответ либо пустой, либо это отрезок, концы которого являются подножеством концов исходных отрезков (убедиться в этом можно, нарисовав все возможные варианты пересечения отрезков на одной прямой).

Получаем такой код пересечения отрезков на одной прямой:

```
static segment::NoSegment() { return segment(pt::NoPt(), pt::NoPt()) };


bool pt::BelongsTo(segment& s) {
  pt A = s.pts[0];
  pt B = s.pts[1];

  pt AB = pt(A, B);
  pt BA = pt(B, A)
  pt AP = pt(A, *this);
  pt BP = pt(B, *this);

  // векторное произведение
  return AB % AP == 0 && BA % BP == 0;
}

pair<bool, segment> IntersectSegmentsOnSameLine(segment& s1, segment& s2) {
  vector<pt> intersection;
  for (int segment_i = 1; segment_i <= 2; ++segment_i) {
    segment&   cur_s = (segment_i == 1 ? s1 : s2);
    segment& other_s = (segment_i == 1 ? s2 : s1);

    for (int i = 0; i < 2; ++i) {
      pt& p = cur_s.ends[i];
      if (p.BelongsTo(other_s)) {
        intersection.push_back(p);
      }
    }
  }

  if (intersection.size() == 0) {
    return {false, segment::NoSegment()};
    // в принципе неважно, что вы тут вернёте вторым аргументом
  }

  return {true, segment(intersection[0], intersection[1])};
}
```

Если же отрезки не лежат на одной прямой, то точек пересечения либо 0, либо 1. Первый способ пересечения отрезков который тогда приходит в голову: найти точку пересечения прямых, на которых лежат отрезки и проверить лежит ли эта точка на каждом из отрезков. Однако коль скоро мы работаем с компьютерами, надо держать в голове проблему с точностью, которая наверняка всплывёт как только мы начнём пересекать прямые $\Longrightarrow$ точка пересечения будет иметь нецелые координаты и проверить её принадлежность каждому из отрезков с абсолютной точностью мы не сможем.

Однако, в случае, когда координаты концов отрезков целочисленные, мы можем наверняка сказать, пересекаются ли отрезки или нет. Для этого должно быть верно следующее: концы каждого из отрезков лежат по разные стороны от прямой, содержащей другой отрезок, либо один из концов отрезков должен лежать на другом отрезке.

Проверку принадлежности точки отрезку мы уже умеем делать, проверять, лежат ли точки по разную сторону от прямой мы тоже умеем (/index.php?title=%D0%9F%D1%80%D1%8F%D0%BC%D1%8B%D0%B5#.D0.9B.D0.B5.D0.B6.D0.B0.D1.82_.D0.BB.D0.B8_.D1.82.D0.BE.D1.87.D0.BA.D0.B8_.D0.BF.D0.B

Автор конспекта: Александр Гришутин

По всем вопросам пишите в telegram @rationalex (https://t.me/rationalex)