|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04 Программная инженерия**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 8 |

**Название:**

Реализация алгоритма отсечения отрезка произвольным выпуклым отсекателем

**Дисциплина:** Компьютерная графика

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-43Б |  | 29.05.2020 | С.С. Кононенко |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | А.В. Куров |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

**1. Цель работы**

Изучение и программная реализация алгоритма отсечения отрезка.

**2. Техническое задание**

Необходимо обеспечить ввод отсекателя – произвольного многоугольника. Высветить его первым цветом. Также необходимо обеспечить ввод нескольких (до десяти) различных отрезков (высветить их вторым цветом). Отрезки могут иметь произвольное расположение: горизонтальные, вертикальные, имеющие произвольный наклон. Предусмотреть ввод отрезков, параллельных границе отсекателя. Ввод осуществлять с помощью мыши и нажатия других клавиш. Выполнить отсечение отрезков, показав результат третьим цветом. Исходные отрезки не удалять.

**3. Теоретический материал**

**Алгоритм Кируса—Бека** — алгоритм отсечения отрезков произвольным выпуклым многоугольником. Был предложен в качестве более эффективной замены алгоритма Коэна—Сазерленда, который выполняет отсечение за несколько итераций.

В данном алгоритме отрезки описываются в форме:

P(t) = P1 + (P2 - P1) \* t, где t - параметр (0 <= t <= 1).

Это векторное уравнение, которое в двумерной графике можно свести к двум одномерным параметрическим уравнениям вида:

Px(t) = P1.x + (P2.x - P1.x) \* t

Py(t) = P1.y + (P2.y - P1.y) \* t, где t - параметр (0 <= t <= 1).

При нахождении пересечений с гранями многоугольника находится значение параметра t, при котором происходит пересечение, и если значение параметра лежит вне допустимых границ, то пересечение происходит с продолжением отрезка. Такие пересечения отвергаются.

В данном алгоритме используется "вектор внутренней нормали" — nв. Вектор внутренней нормали — вектор, перпендикулярный грани многоугольника и направленный внутрь этого многоугольника. Этот факт проверяется аналитическим выражением вида nв \* (B - A) >= 0, где А — точка грани, из которой исходит данная нормаль, а В — любая другая точка нормали.

Рассмотрим вектор — [P(t) - fi], где fi - произвольная точка рассматриваемой грани. Проанализируем скалярное произведение этого вектора и вектора нормали к рассматриваемой грани: nв[P(t) - fi] > 0 — вектор направлен внутрь области многоугольника; nв[P(t) - fi] = 0 — вектор перпендикулярен нормали; nв[P(t) - fi] < 0—направлен вне области многоугольника. При различных параметрах t, точка P(t) может находиться как внутри, так и вне области многоугольника относительно рассматриваемой грани. В данном случае нас больше интересует тот факт, что мы можем определить "входит" или "выходит" отрезок из многоугольника при пересечении определенной грани. Если отрезок пересекает грань и его начало было внутри многоугольника относительно этой грани (анализ скалярного произведения, рассмотренного выше, при P(t), t = 0), то получается что при пересекании он выйдет за грань. Рассмотрим ситуацию nв[P(t) - fi] = 0. Она соответствует ситуации, когда вектор, состоящий из точки отрезка (P(t)) и точки fi грани параллелен этой грани. Если стоит задача построить параллельную прямую к некоторой прямой L через точку А, находящейся на этой прямой L, то решение этой задачи — прямая, совпадающая с прямой L. Из этого следует, что nв[P(t) - fi] = 0 выполняется тогда, когда вектор лежит на одной прямой с гранью [P(t) - fi], а P(t) для некоторого t — точка пересечения грани и отрезка.

Подставим параметрическую форму уравнения в выражение   
nв[P1 + (P2 - P1)t - fi] = 0

nв[P1 - fi] + nв[P2 - P1]t = 0

В данном уравнении [P2 - P1] - вектор, определяющий направление отрезка, а   
[P1 - fi] — вектор, соединящий некоторую точку грани с началом отрезка.

Введем обозначения:

Wi = nв[P1 - fi]

Dск = nв[P2 - P1]

Выразим t:

t = -Wi / Dск

Данное выражение нельзя рассматривать при Dск = 0. Рассмотрим случаи, когда Dск = 0:

1. Вектор ориентации отрезка вырожден. Такой случай нас не интересует.
2. Dcк перпендикулярен nв. Отрезок парллелен грани. Здесь может быть 2 случая: отрезок лежит вне многоугольника относительно грани - тогда можно однозначно сделать вывод о том, что он невидим и закончить операцию отсечения данного отрезка. Отрезок лежит внутри многоугольника относительно грани - тогда следует перейти на следующую итерацию и продолжить операцию отсечения. Определить "вне" или "внутри" довольно-таки легко с помощью вектора Wi. Этот вектор начинается в некоторой точке рассматриваемой грани многоугольника и заканчивается в некоторой точке отрезка, таким образом можно сказать, что он направлен "от грани к отрезку". Внутренняя нормаль начинается в некоторой точке грани и может быть направлена к отрезку и в противоположную сторону. Проверяется это так Wi = nв[P1 - fi]. Если скалярное произведение < 0, то угол между вектором нормали и вектором, направленным к отрезку > 90, и вектор лежит вне фигуры, иначе - внутри.

Если отрезок виден, то он виден относительно всех граней. Из этого можно сделать вывод, что если отрезок входит в многоугольник относительно какой-то грани, то относительно других граней он должен был уже войти, и при этом не должен был выйти. В случае «выхода»: выйдя за первую грань, отрезок перестанет быть виден относительно нее и, следовательно, будет на оставшемся промежутке. Далее, убедившись, что параметр tвх , соответствующий последнему входу, меньше, чем параметр tвых, соответствующий первому выходу, чертим видимую часть отрезка (если условие не выполняется - не чертим).

**4. Реализация алгоритма**

**def** **cyrusbeck**(root, cut, section, normals):

t\_start = **0**

t\_end = **1**

vect = get\_vect(section[**0**], section[**1**])

cutlen = len(cut)

**for** i **in** range(cutlen):

w\_vect = get\_vect(cut[(i + **1**) % cutlen], section[**0**])

**if** cut[i] != section[**0**]:

w\_vect = get\_vect(cut[i], section[**0**])

vect\_scal = get\_scalar\_mul(vect, normals[i])

w\_vect\_scal = get\_scalar\_mul(w\_vect, normals[i])

**if** vect\_scal == **0**:

**if** w\_vect\_scal < **0**:

**return**

**continue**

t = -w\_vect\_scal / vect\_scal

**if** vect\_scal > **0**:

**if** t > t\_start:

t\_start = t

**else**:

**if** t < t\_end:

t\_end = t

**if** t\_start > t\_end:

**break**

**if** t\_start < t\_end:

dot\_start = [round(section[**0**][**0**] + vect[**0**] \* t\_start),

round(section[**0**][**1**] + vect[**1**] \* t\_start)]

dot\_end = [round(section[**0**][**0**] + vect[**0**] \* t\_end),

round(section[**0**][**1**] + vect[**1**] \* t\_end)]

root.draw\_line(dot\_start, dot\_end, root.res\_color)

Реализация вспомогательных функций (операции над векторами, отсекателем, нормалями):

**def** **get\_vect**(dot\_start, dot\_end):

**return** [dot\_end[**0**] - dot\_start[**0**], dot\_end[**1**] - dot\_start[**1**]]

**def** **get\_vect\_mul**(fvector, svector):

**return** fvector[**0**] \* svector[**1**] - fvector[**1**] \* svector[**0**]

**def** **get\_scalar\_mul**(fvector, svector):

**return** fvector[**0**] \* svector[**0**] + fvector[**1**] \* svector[**1**]

**def** **get\_normal**(dot\_start, dot\_end, dot\_check):

vect = get\_vect(dot\_start, dot\_end)

normal = None

**if** vect[**0**] == **0**:

normal = [**1**, **0**]

**else**:

normal = [-vect[**1**] / vect[**0**], **1**]

**if** get\_scalar\_mul(get\_vect(dot\_end, dot\_check), normal) < **0**:

**for** i **in** range(len(normal)):

normal[i] = -normal[i]

**return** normal

**def** **get\_normals**(cut):

normals = []

cutlen = len(cut)

**for** i **in** range(cutlen):

normals.append(get\_normal(cut[i], cut[(i + **1**) % cutlen], cut[(i + **2**) % cutlen]))

**return** normals

**def** **check\_cut**(cut):

**if** len(cut) < **3**:

**return** False

vect1 = get\_vect(cut[**0**], cut[**1**])

vect2 = get\_vect(cut[**1**], cut[**2**])

sign = None

**if** get\_vect\_mul(vect1, vect2) > **0**:

sign = **1**

**else**:

sign = -**1**

**for** i **in** range(**3**, len(cut)):

vecti = get\_vect(cut[i-**2**], cut[i-**1**])

vectj = get\_vect(cut[i-**1**], cut[i])

**if** sign \* get\_vect\_mul(vecti, vectj) < **0**:

**return** False

**if** sign < **0**:

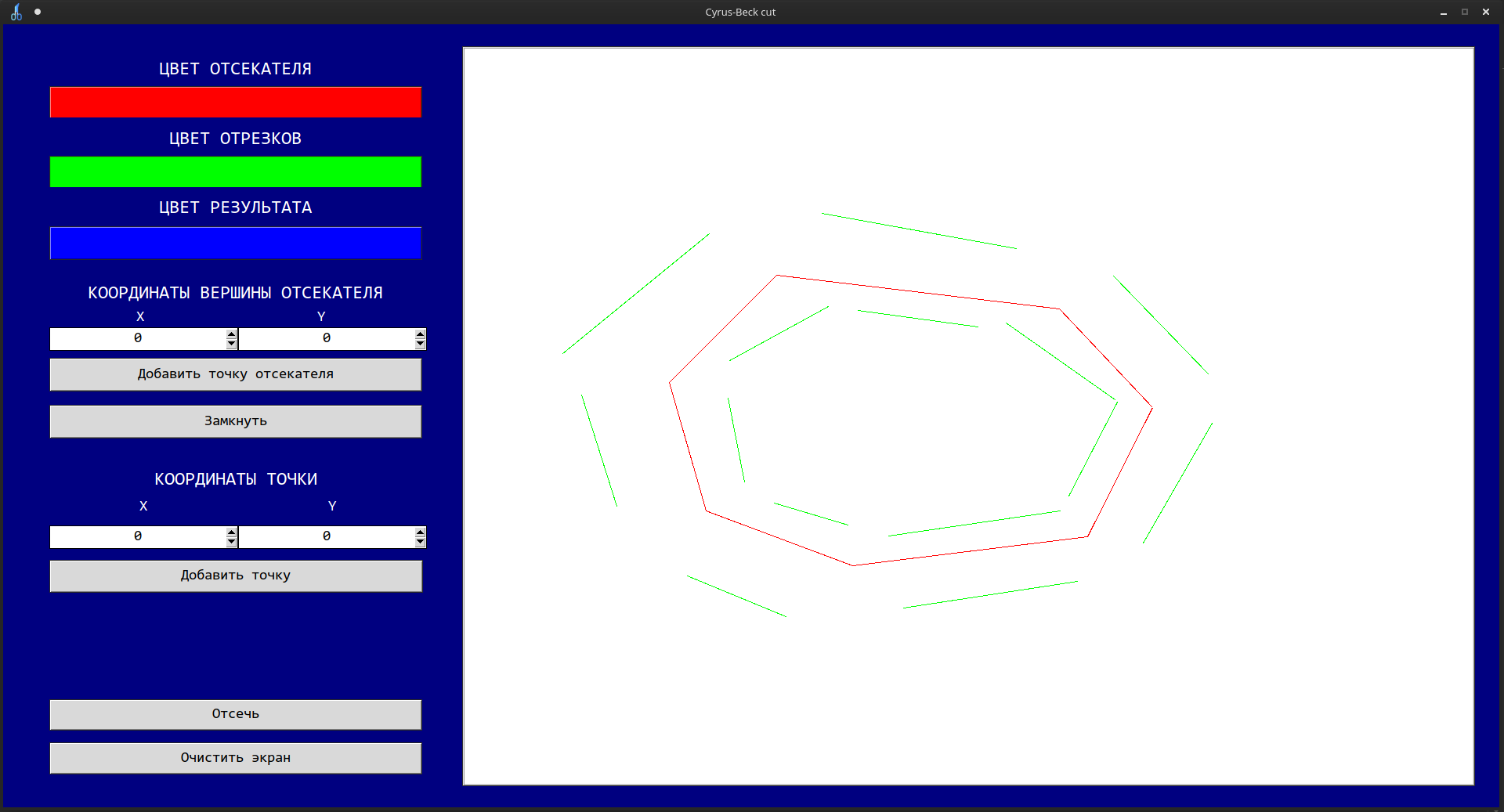
cut.reverse()

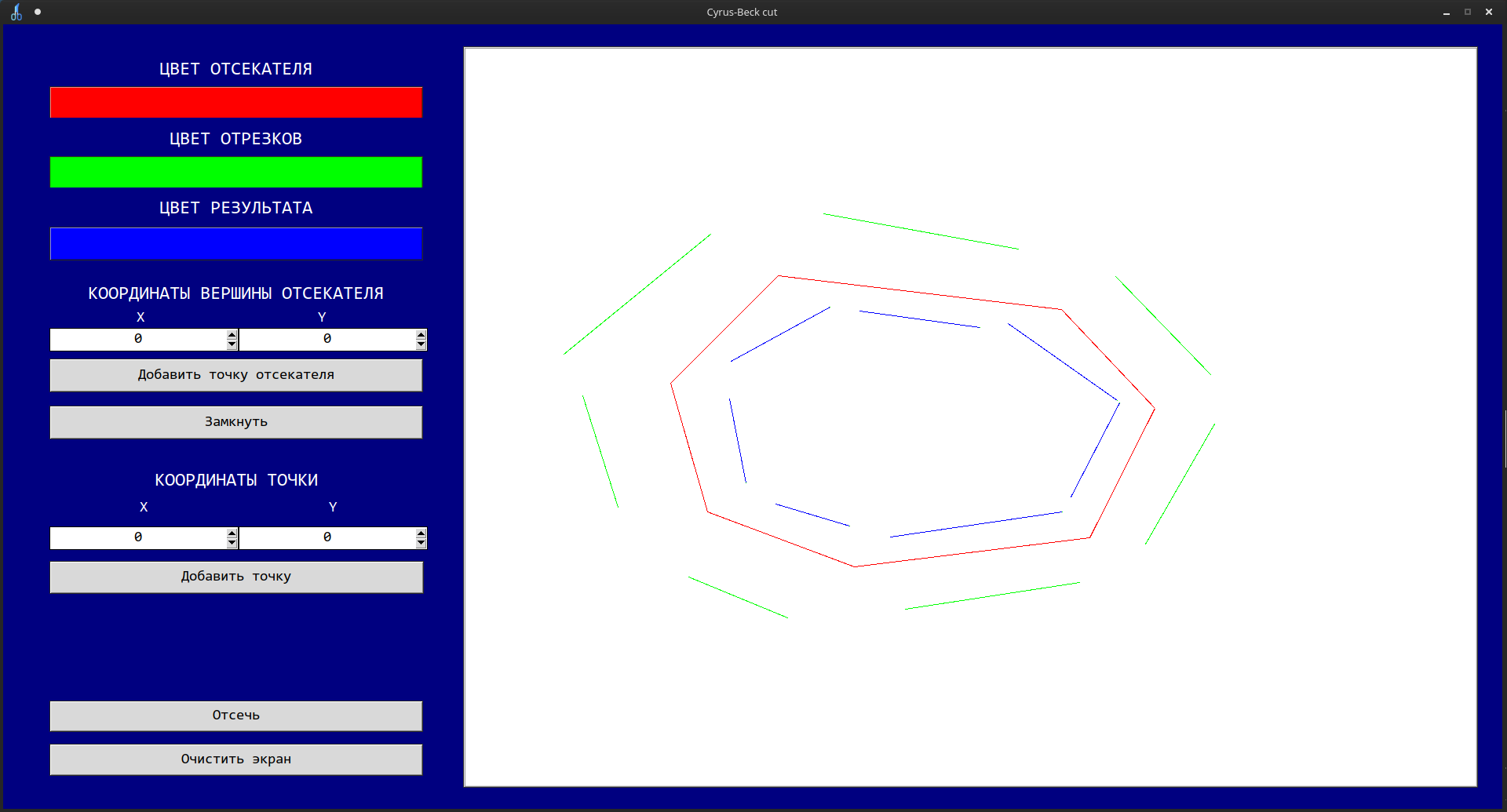
**return** True

**5. Демонстрация работы программы**

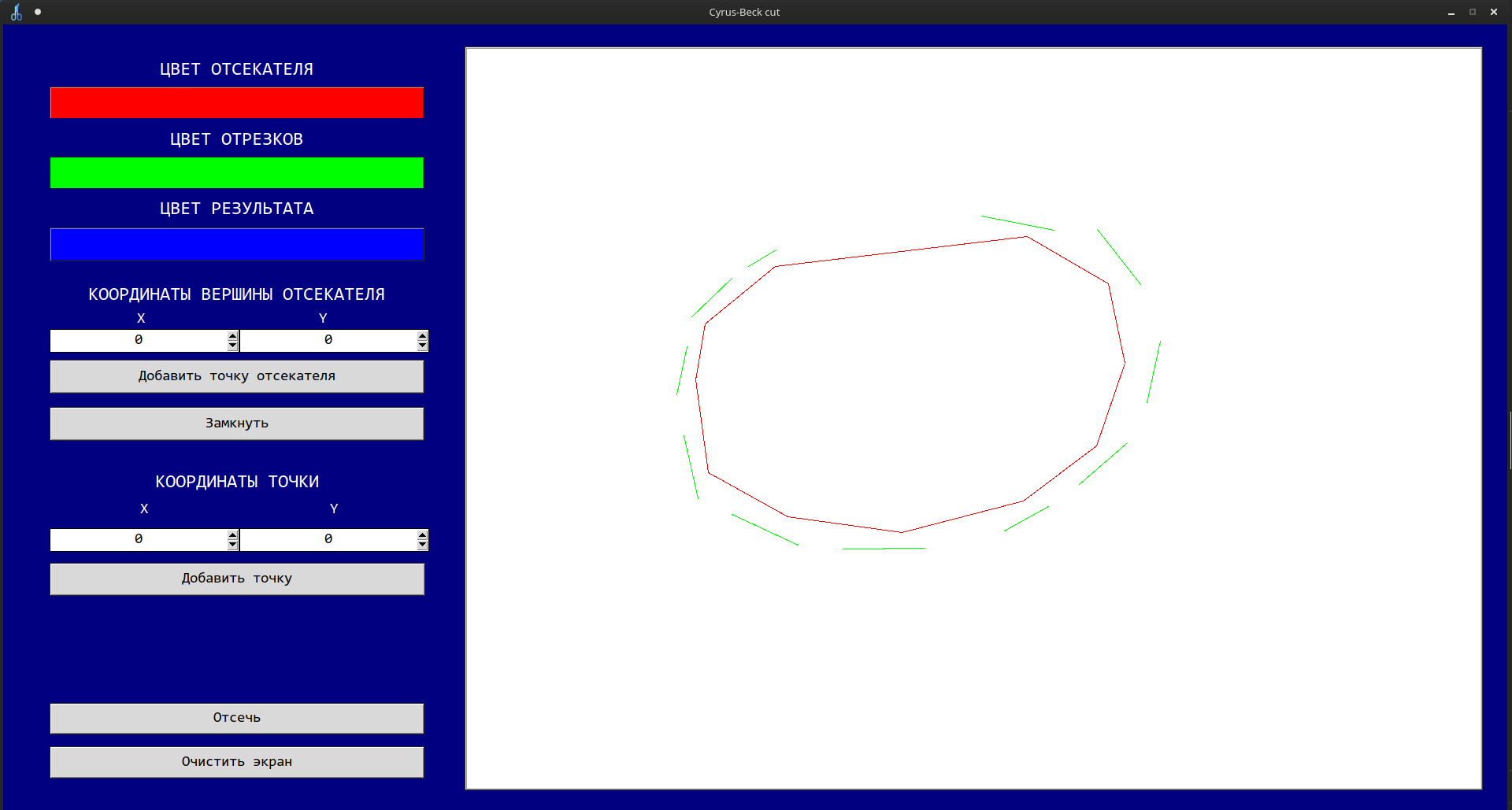
Тривиальные случаи.

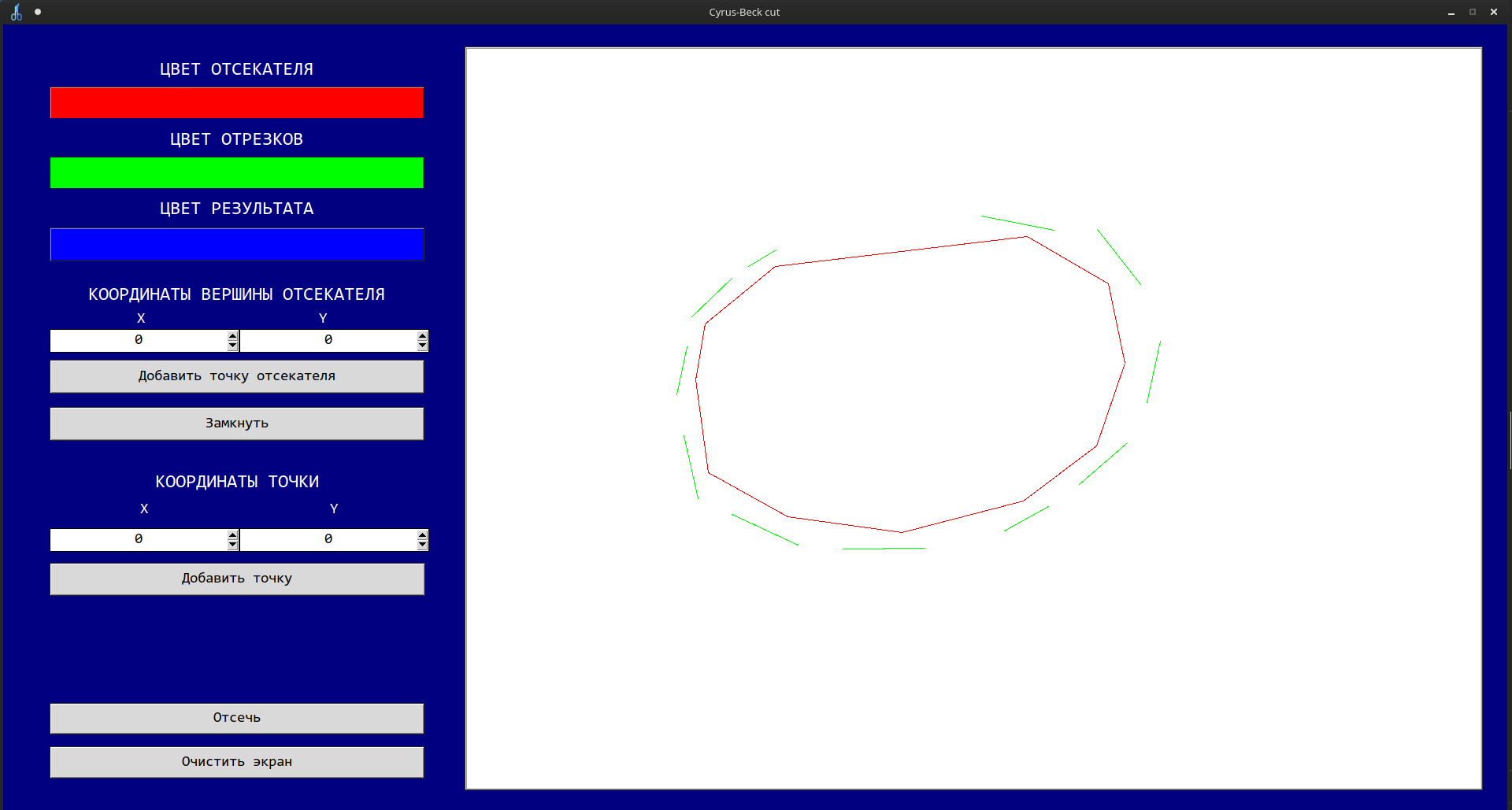
До:

После:



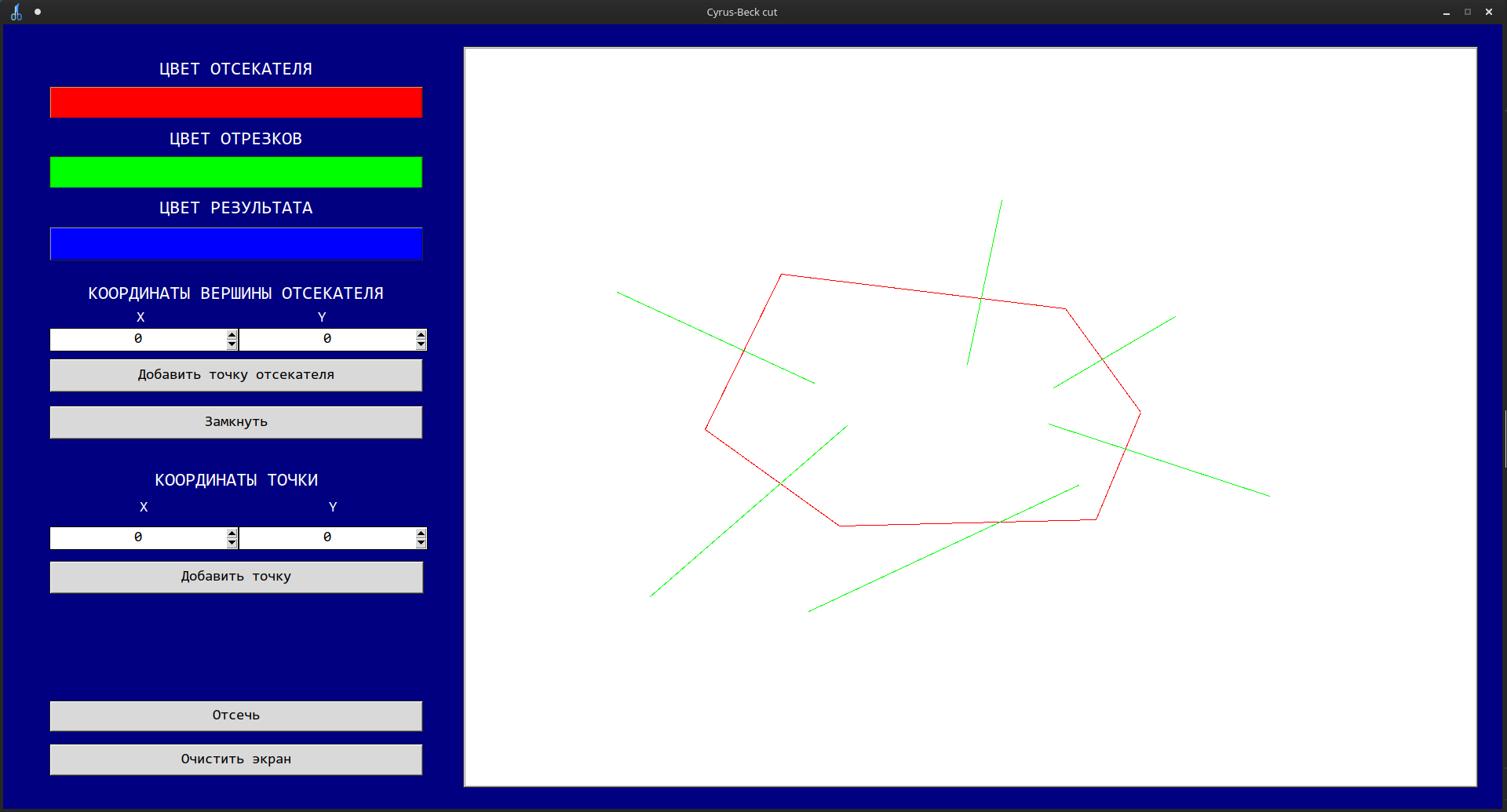
До:

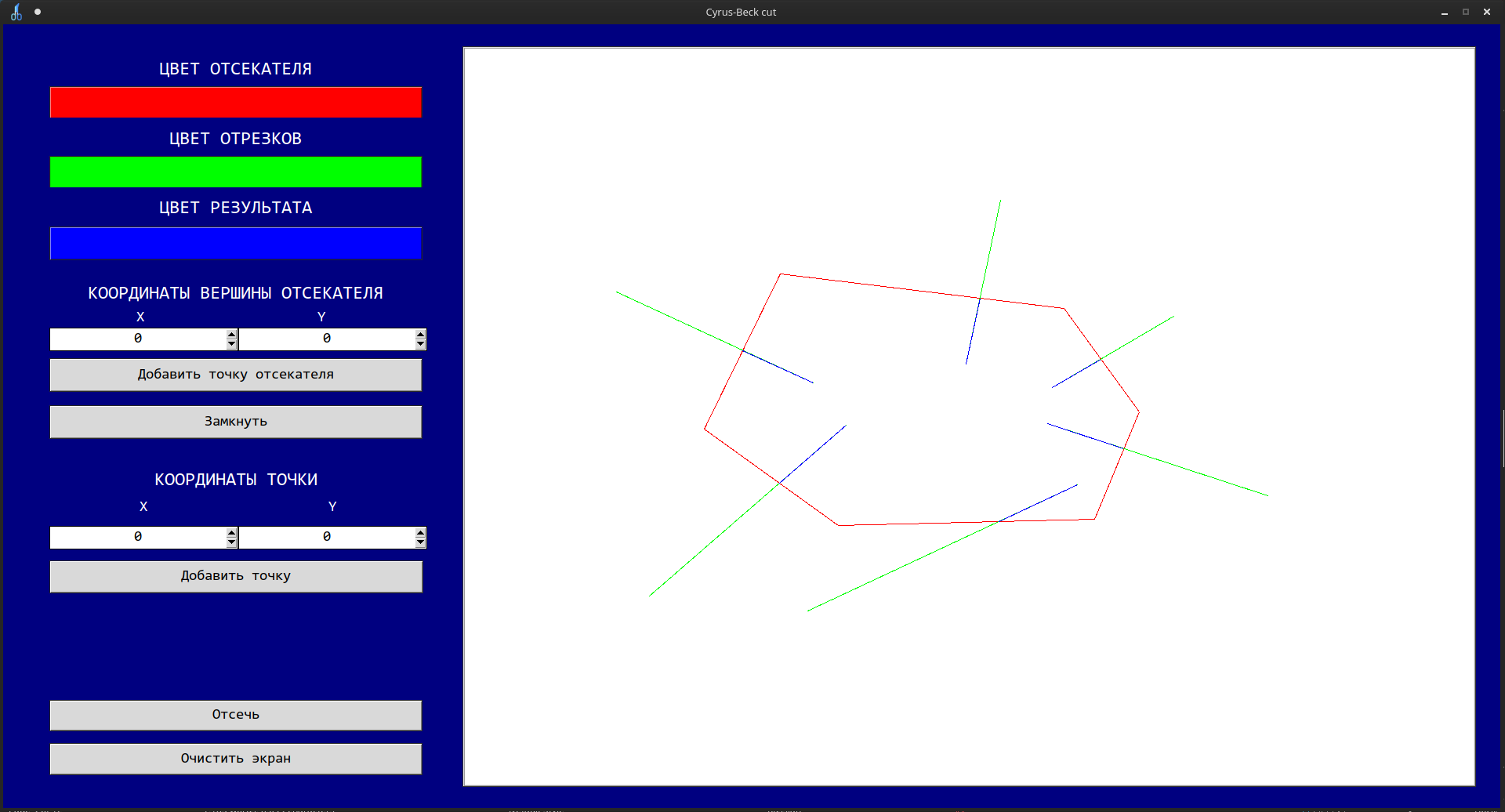
После:



Пересечение отрезка и отсекателя в одной точке.

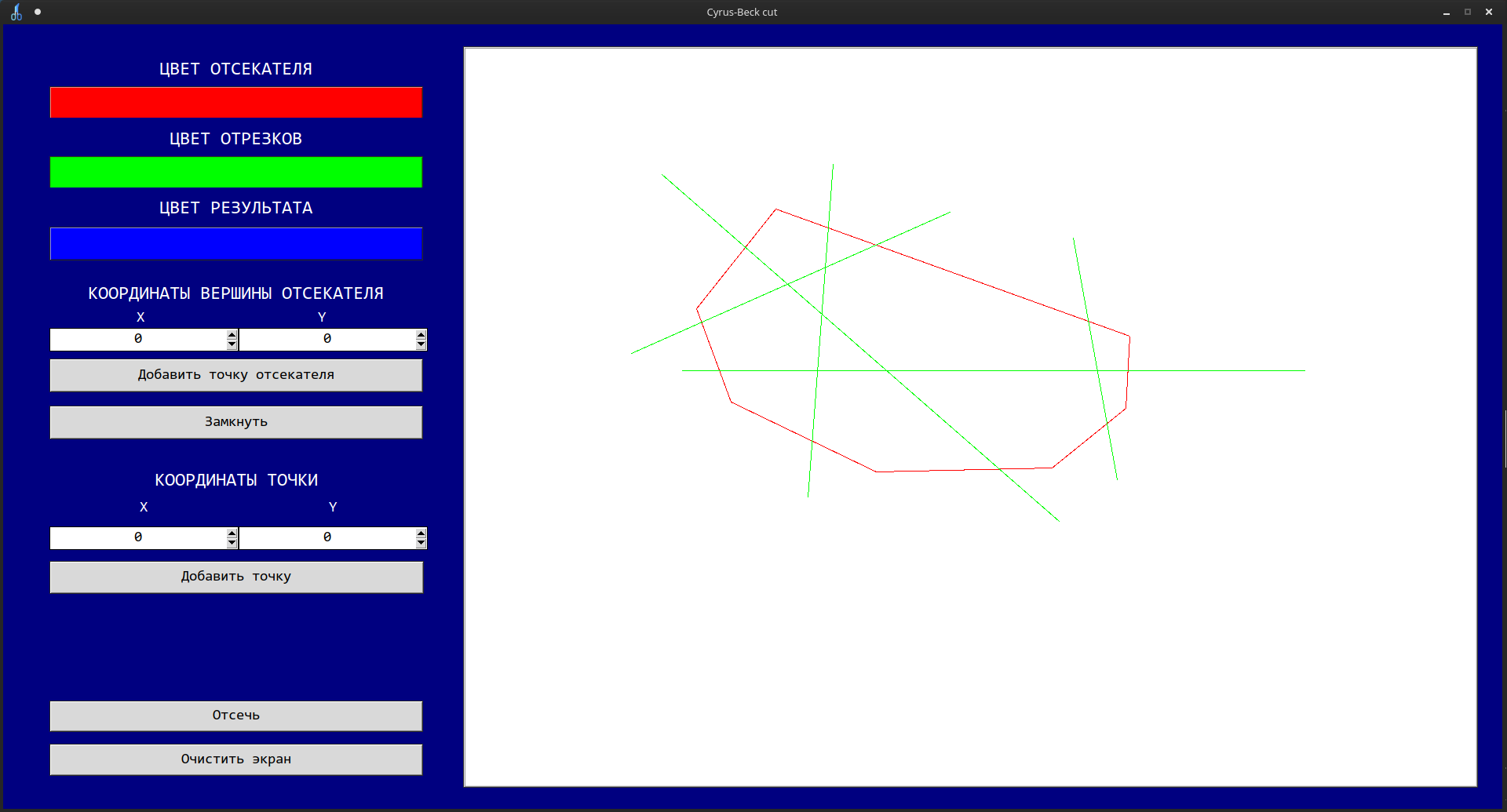
До:

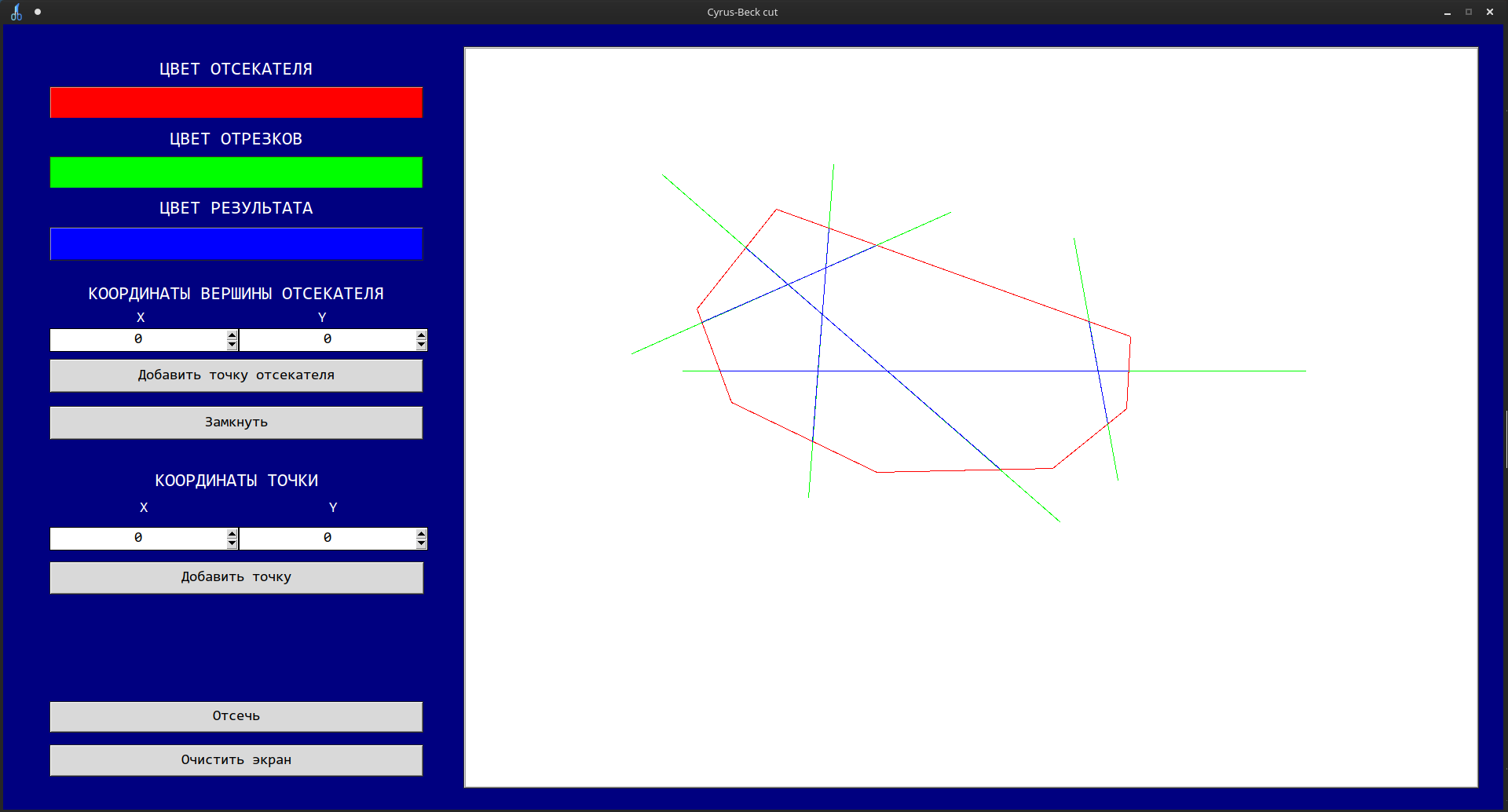
После:



Пересечение отрезка и отсекателя в двух точках.

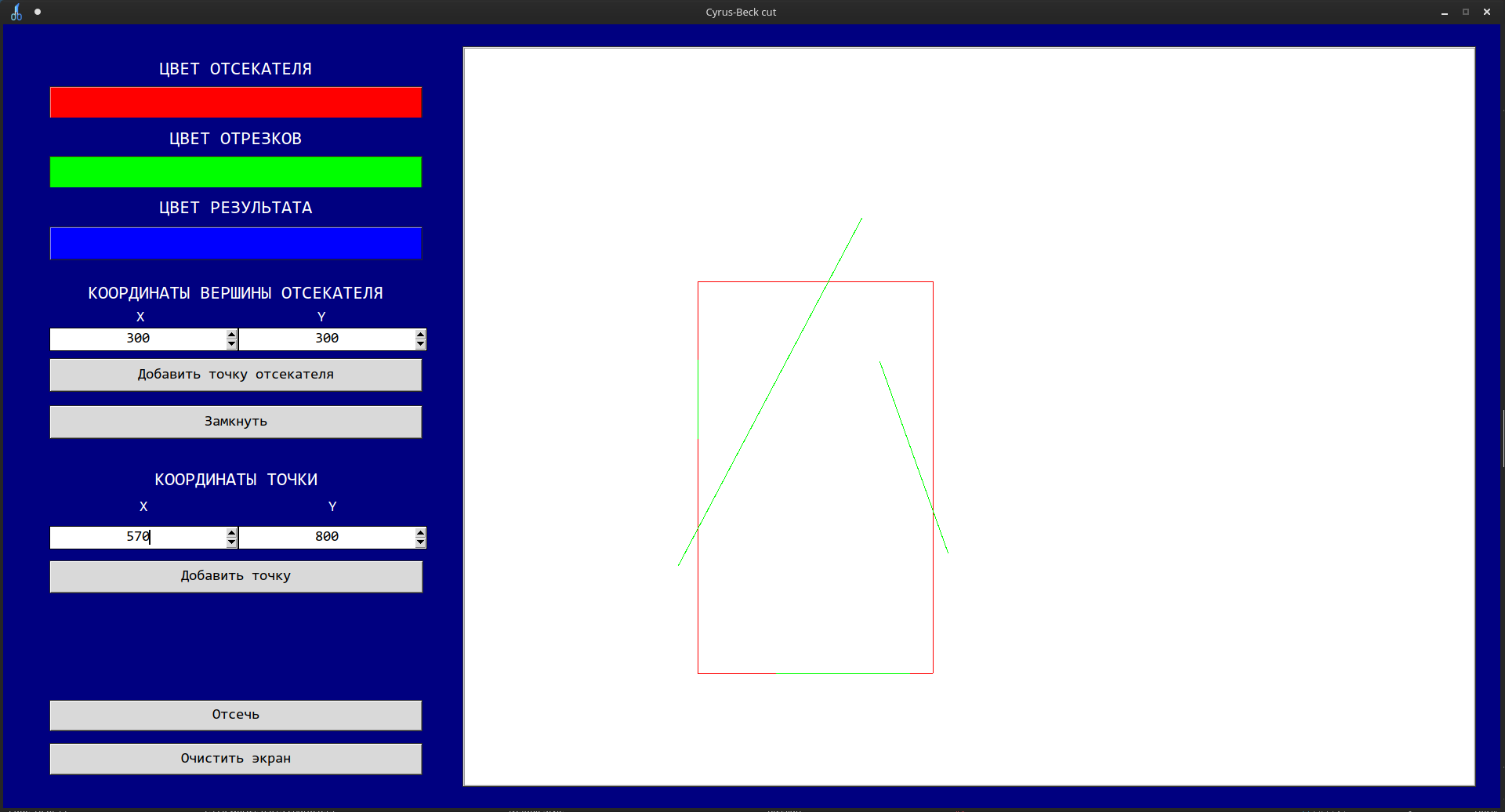
До:

После:



Отрезки располагаются на границах отсекателя.

До:

После:

