|  |  |
| --- | --- |
| **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»

**Лабораторная работа №5**

**по дисциплине «Компьютерная графика»**

|  |  |
| --- | --- |
| **Тема:** Реализация и исследование алгоритмов растрового заполнения сплошных областей.  **Студент:** Пересторонин Павел  **Группа:** ИУ7-43Б  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель:** Куров А. В. |  |

Москва.

2020 г.

**Цель работы:**

Реализация и исследование одного из алгоритмов (по заданию преподавателя) растрового заполнения области.

**Задание:**

Реализовать **алгоритм заполнения со списком ребер и флагом**. Обеспечить ввод произвольной многоугольной области, содержащей произвольное количество отверстий.

Работа программы должна предусматривать два режима – с задержкой и без задержки.

Обеспечить замер времени выполнения алгоритма (без задержки, с выводом на экран только окончательного результата).

**Теоретический материал**

*Растровая развертка сплошных областей* – генерация сплошных областей из простых описаний ребер или вершин. В методах растровой развертки пытаются определить в порядке сканирования строк, лежит ли точка внутри многоугольника или контура. Алгоритмы обычно идут от «верха» многоугольника к «низу».

**Алгоритм со списком ребер и флагом.**

Алгоритм, использующий список ребер и флаг, является двухшаговым. Первый шаг состоит из обрисовки контура, в результате которой на каждой сканирующий строке образуются пары ограничивающих пикселов. Второй шаг состоит в заполнении пикселов, расположенных между ограничивающими.

Первый шаг (нахождение и пометка флагов):

Найдем и отметим для каждого ребра все его пересечения со сканирующими строками. Для каждого пересечения со сканирующей строкой будем отмечать ближайший к пересечению *правый* пиксел. В дальнейшем эти пикселы будут “флагом смены цвета” (признаком того, что нужно сменить фоновый цвет цветом закраски или наоборот).

*Примечание: тот факт, что всегда выбирается правый пиксел обусловлен следующими соображениями:*

*1.* ***Случай, когда мы идем по сканирующей строке вне фигуры и заходим в нее (в данном случае мы идем слева направо).*** *Если ребро проходит левее ближайшего пиксела, то будет выбран этот ближайший пиксел (потому что он будет ближайшим правым). Если же ребро проходит правее центра ближайшего пиксела, то фигуре принадлежит меньше половины площади этого пиксела. Поэтому алгоритм, дойдя до пересечения, этот пиксел не закрасит, а следующий, который будет помечен, закрасит (потому что при встрече флага мы сразу меняем цвет и окрашиваем уже новым цветом).*

*2.* ***Случай, когда мы идем по сканирующей строке внутри фигуры и выходим из нее (в данном случае мы опять же идем слева направо).*** *Если ребро проходит левее центра, то получается фигуре принадлежит меньшая по площади часть пиксела. Учитывая тот факт, как мы выбираем пиксел (ближайший правый), этот пиксел будет выбран и при наступании на него цвет поменяется на фоновый. Если же центр пиксела левее пересечения, то значит бОльшая часть площади принадлежит фигуре и цвет нужно сменить начиная только со следующего пиксела. Выбирая* ***правый*** *от пересечения пиксел у нас так и выйдет.*

Второй шаг (закраска):

На данном шаге для каждой сканирующей строки мы действуем по следующему принципу:

1. Если встречаем отмеченный пиксел, то меняем состояние (внутри/снаружи), либо цвет (закраски/фона)

2. Помечаем пиксел текущим цветом.

Примечание: интервал для х и у берется исходя из минимального окаймляющего прямоугольника.

Псевдокод:

Для **у** в интервале **[ymin; ymax]**

**{**

**Цвет = фоновый**

для **х** в интервале **[xmin; xmax]**

**{**

если пиксел в точке **х** имеет граничное значение

**{**

тогда инвертировать **Цвет**

**}**

окрасить пиксел в точке (**x**, **y**) цветом **Цвет**

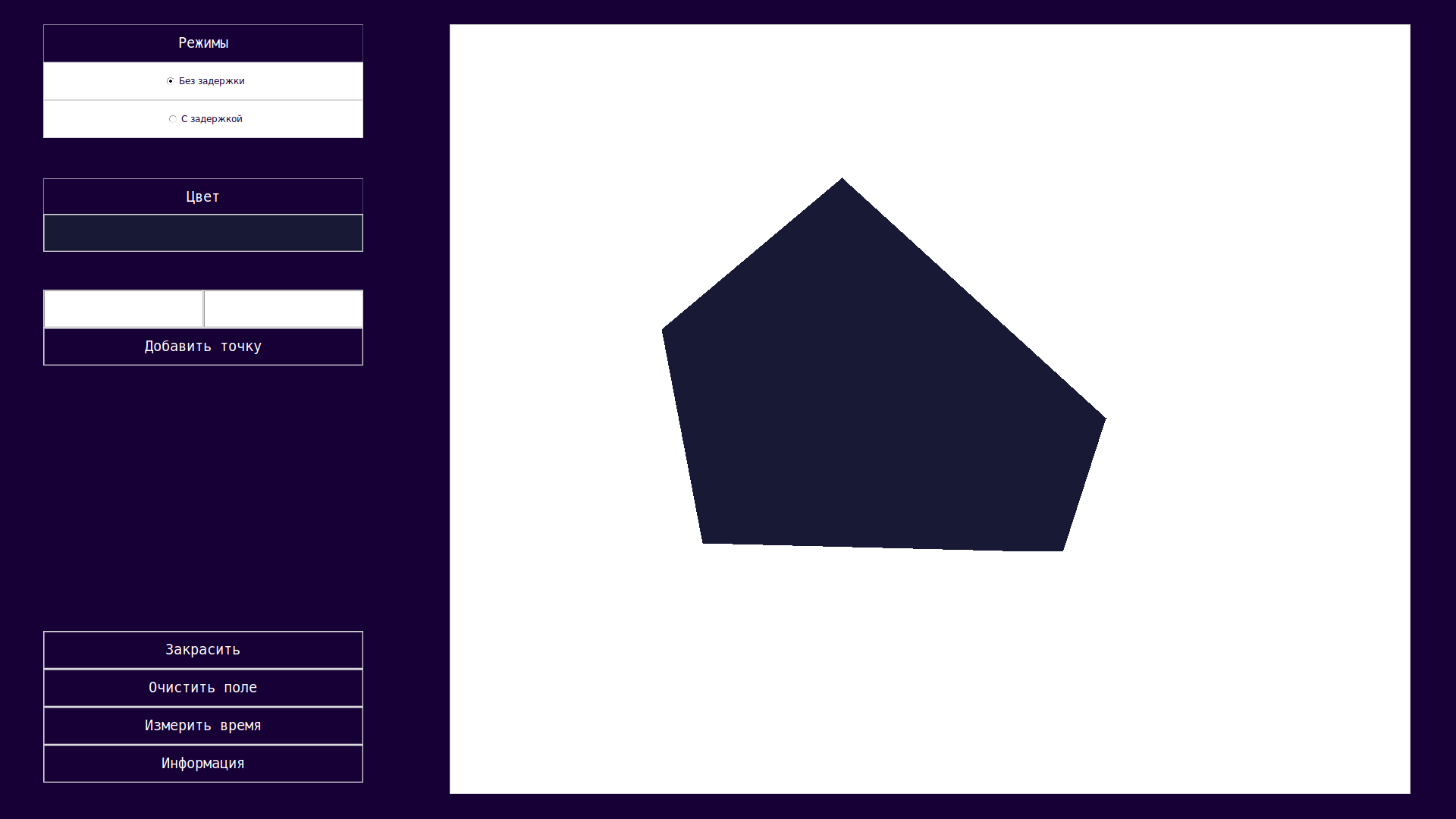
**}**

**}**

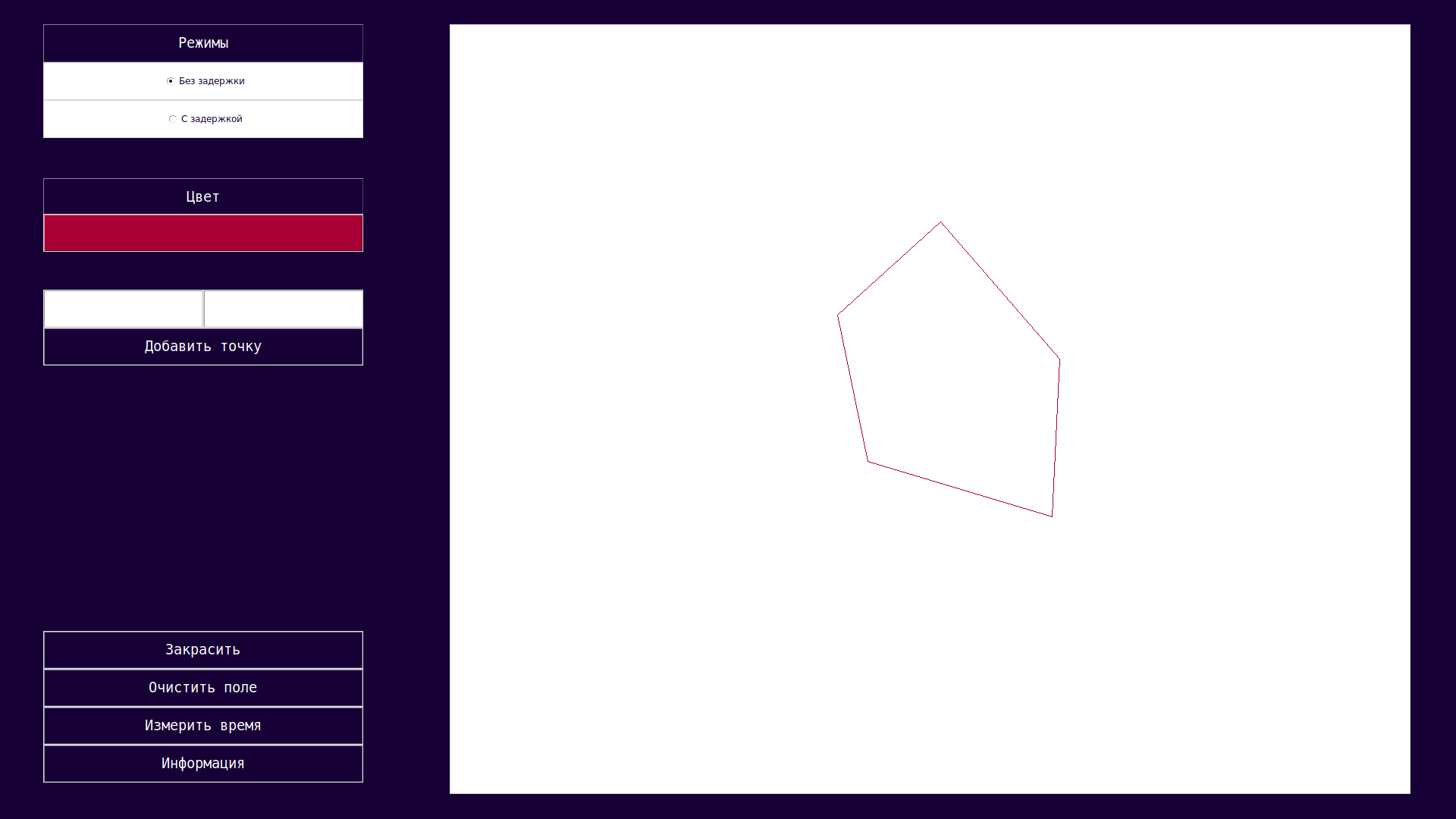
**Исходный код. Примеры работы программы.**

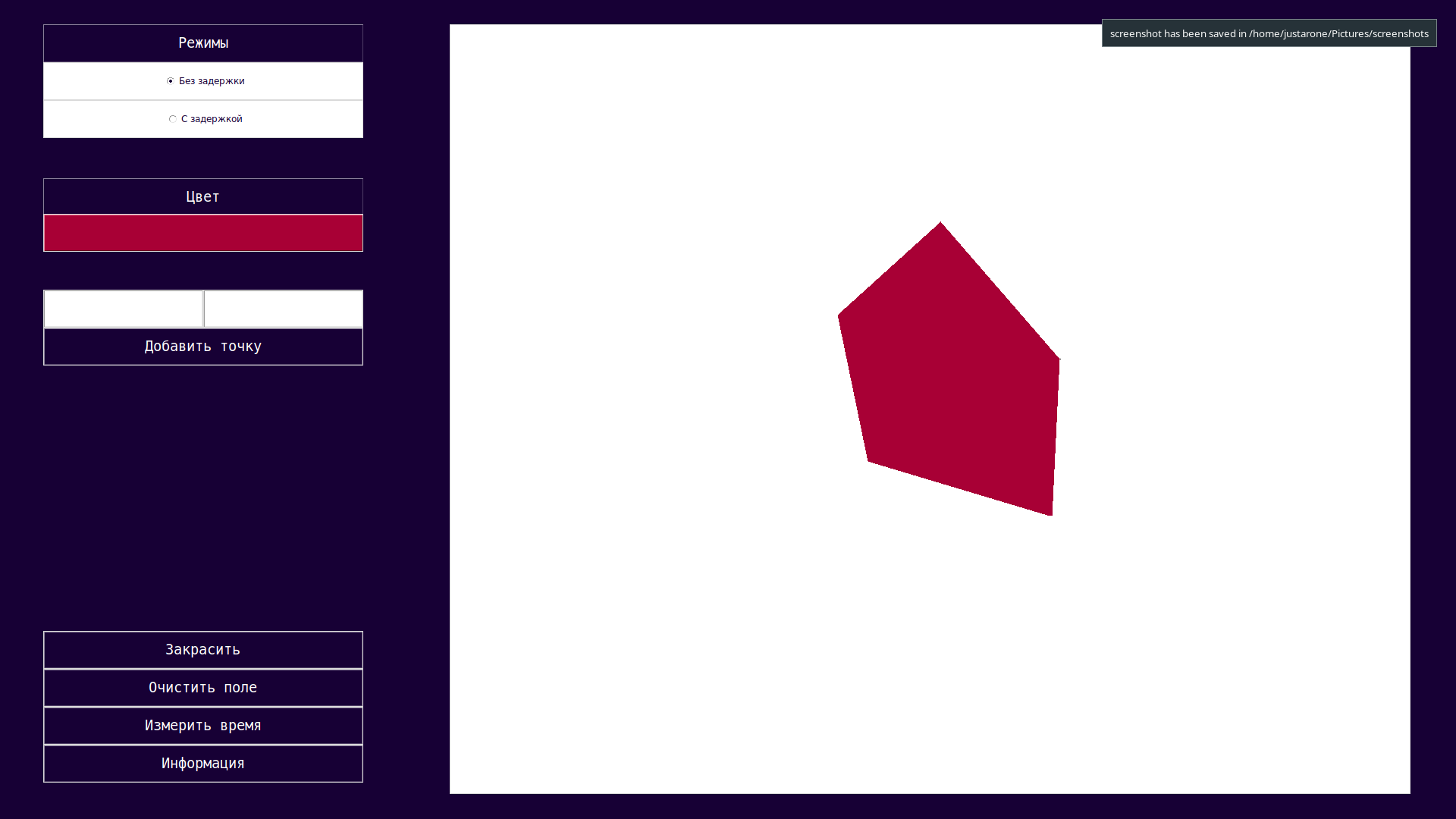
****

Интерфейс:

**

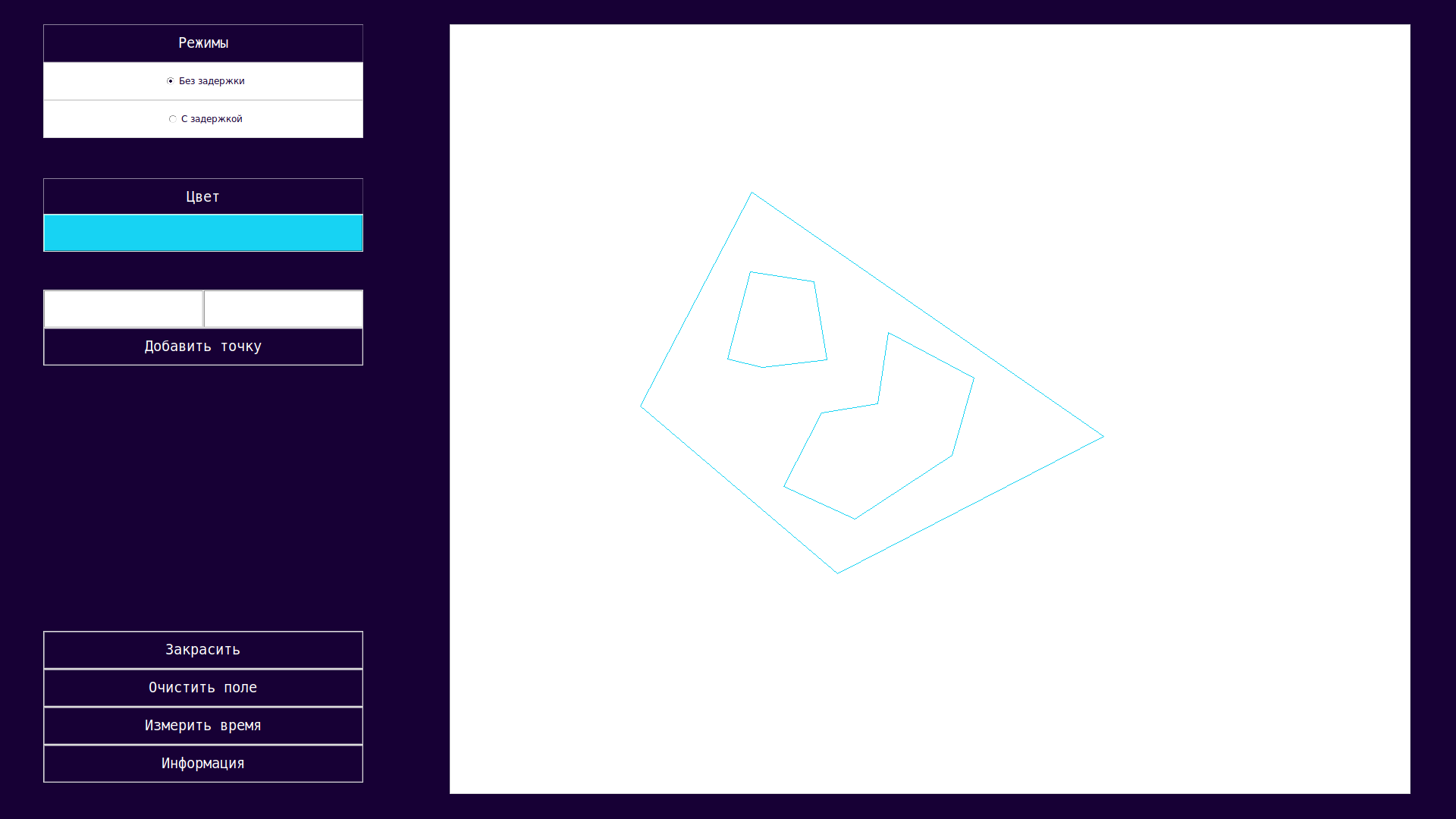
Заполнение произвольного многоугольника:

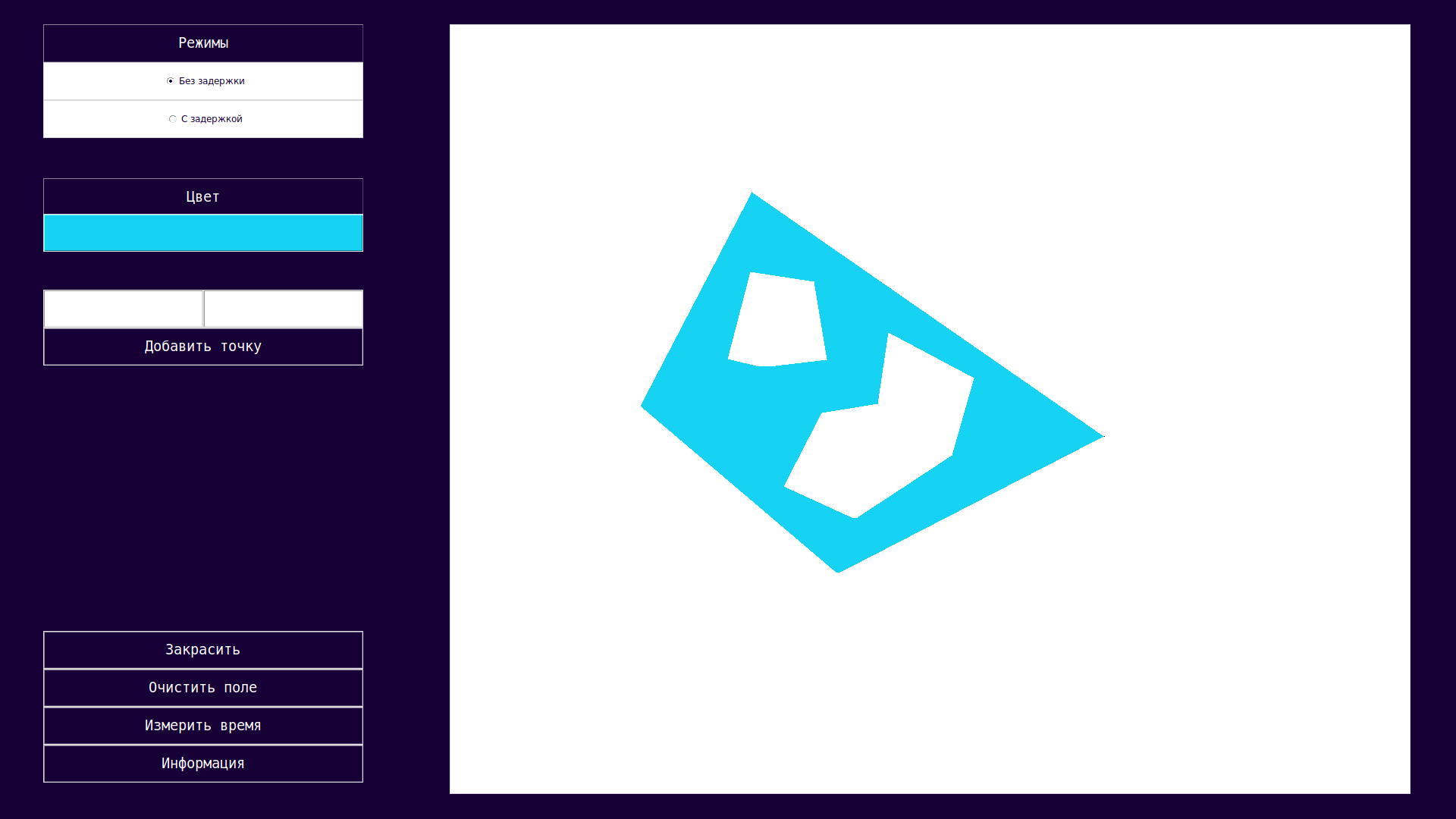




Время заполнения: 0.093424 секунды.

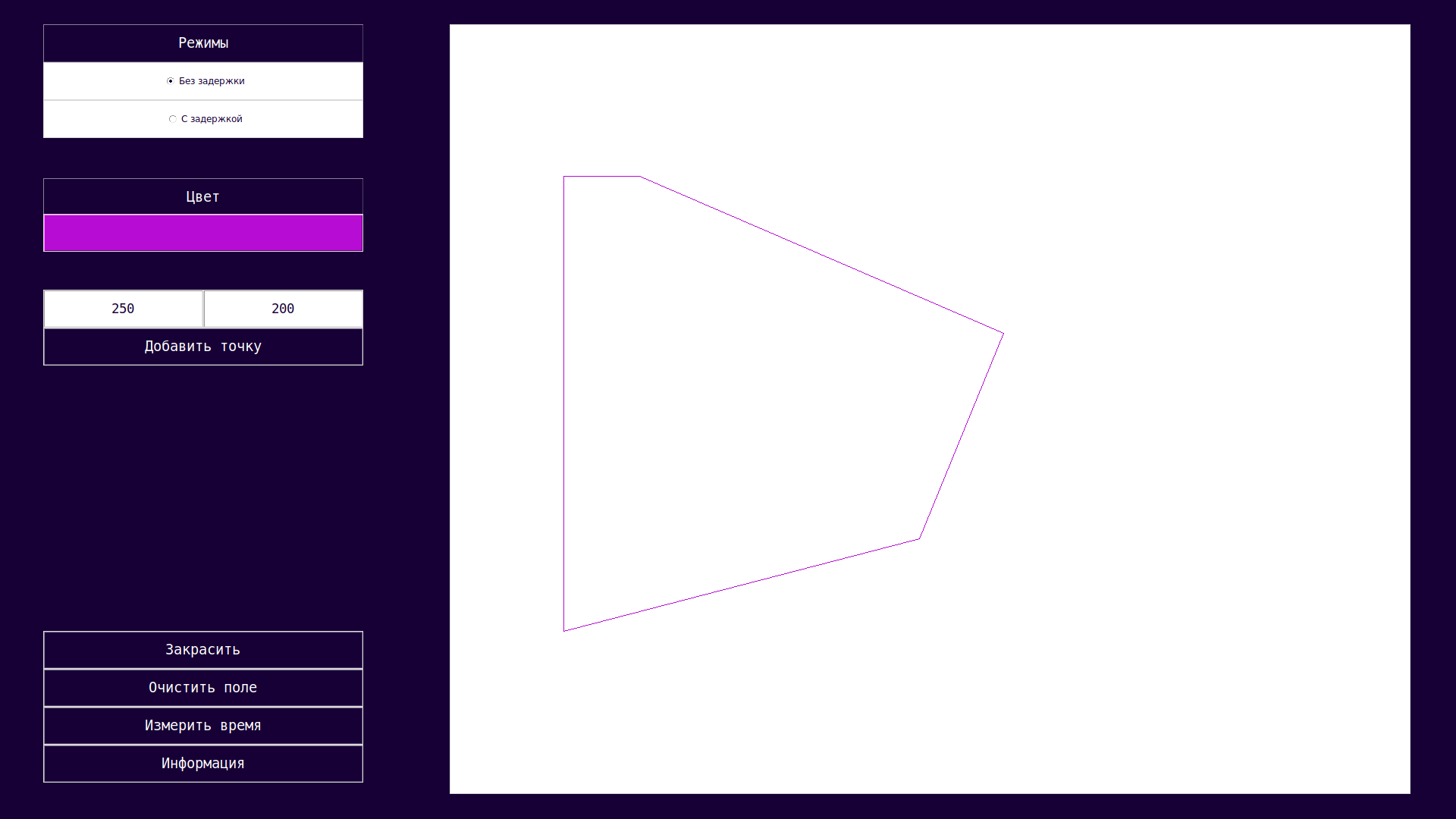
Заполнение произвольного многоугольника с отверстиями:

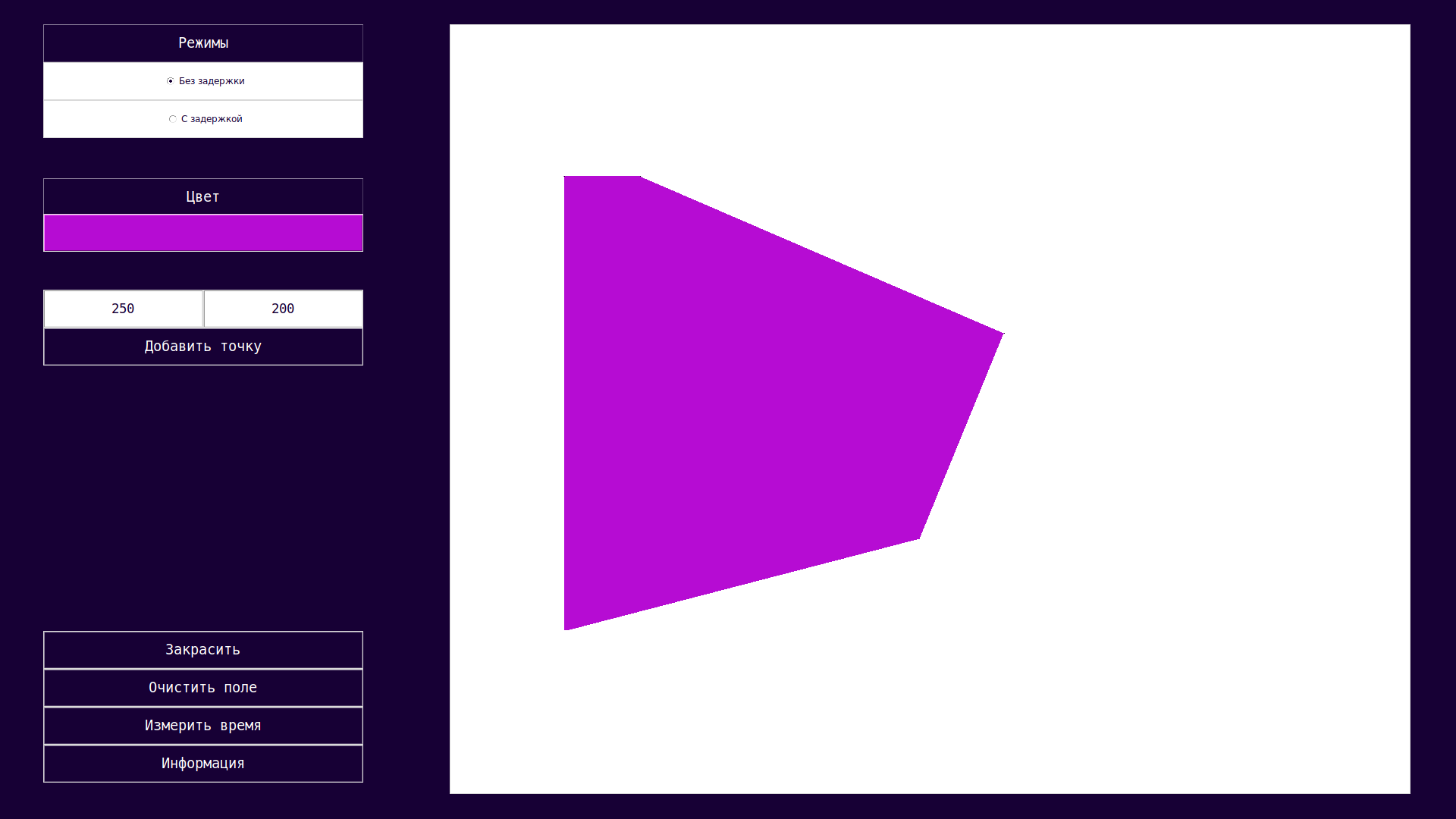




Время заполнения: 0.111983 секунды.

Заполнение многоугольника с вертикальными и горизонтальными прямыми:

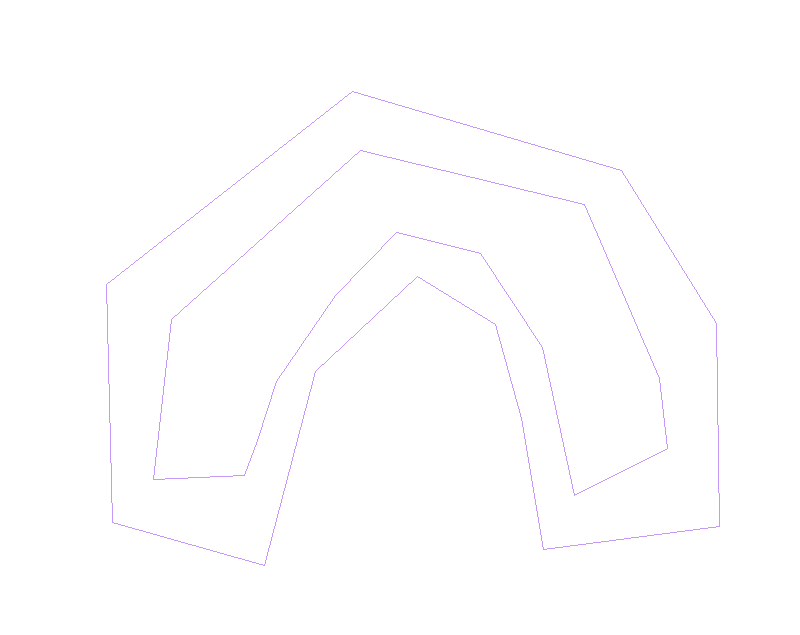




Время заполнения: 0.1260235 секунды.

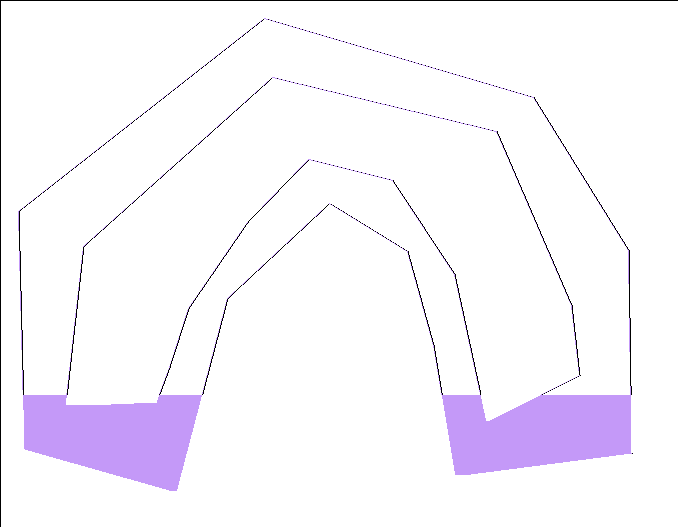
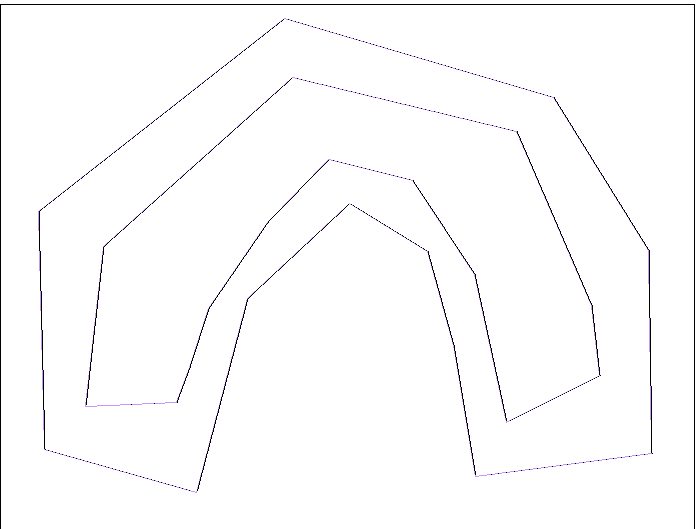
Заполнение произвольного многоугольника с задержкой (2 этапа + доп. Скриншоты во время 2 этапа):

Начальная фигура:



Конец 1 этапа, Флаги расставлены черным цветом (в программе есть проверка на цвет закраски и контура: если они черные, то цвет флага меняется)

Качество не очень хорошее, однако можно заметить, что у более горизонтальных ребер меньше флагов (они менее черные) в связи с меньшим кол-вом пересечений со сканиирующими строками.



Заполнение в процессе (несколько скриншотов):

