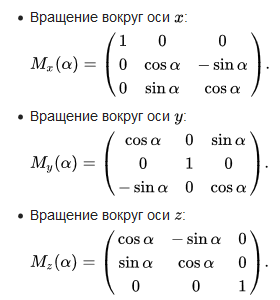
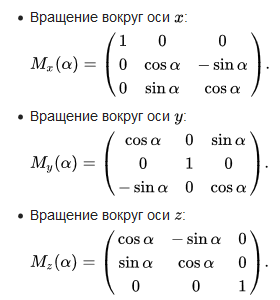
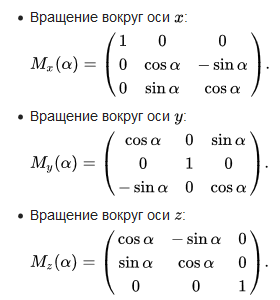
**1. Исследование математических методов представления и преобразования графических объектов на плоскости и в пространстве**

**Задание:** поворот объемного тела относительно осей координат на заданный угол.

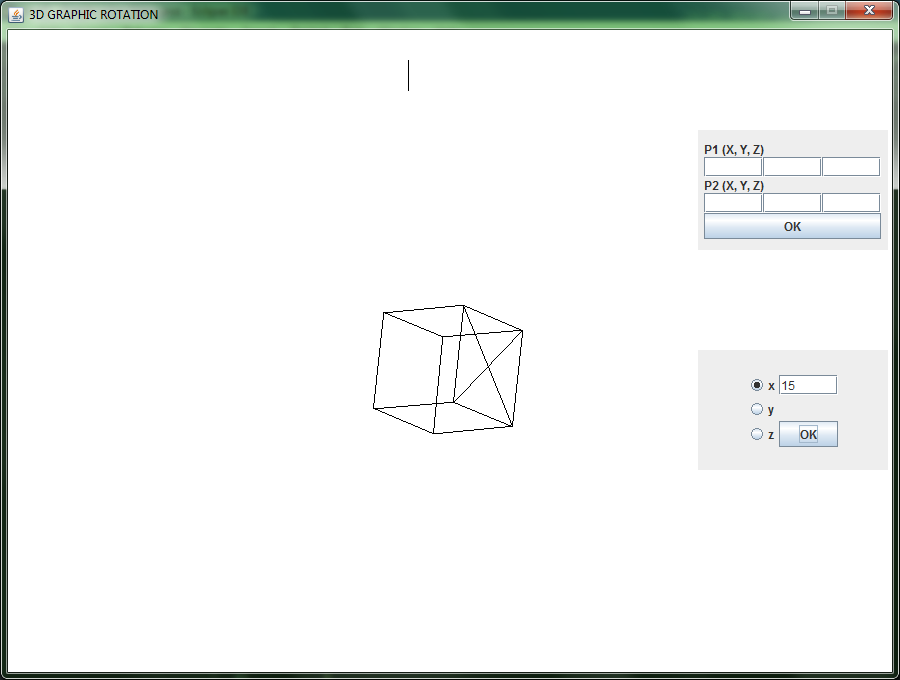
**Теоретические сведения:**



**Пример реализации алгоритма:**



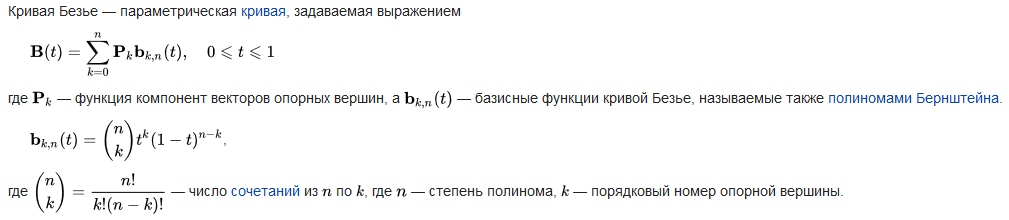
**Скриншот работы программы:**



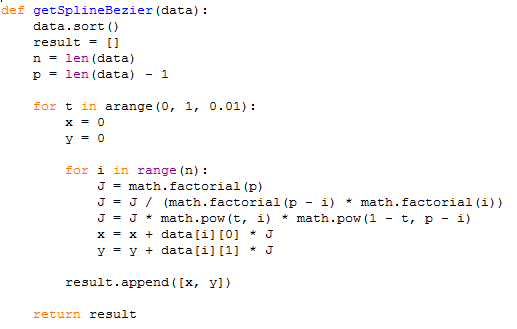
**2. Алгоритмы представления и формирования параметрических кривых.**

**Задание:** сформировать на плоскости кривую Безье на основе задающей ломаной, определяемой 3 и большим количеством точек. Обеспечить редактирование координат точек задающей ломаной с перерисовкой сплайна Безье.

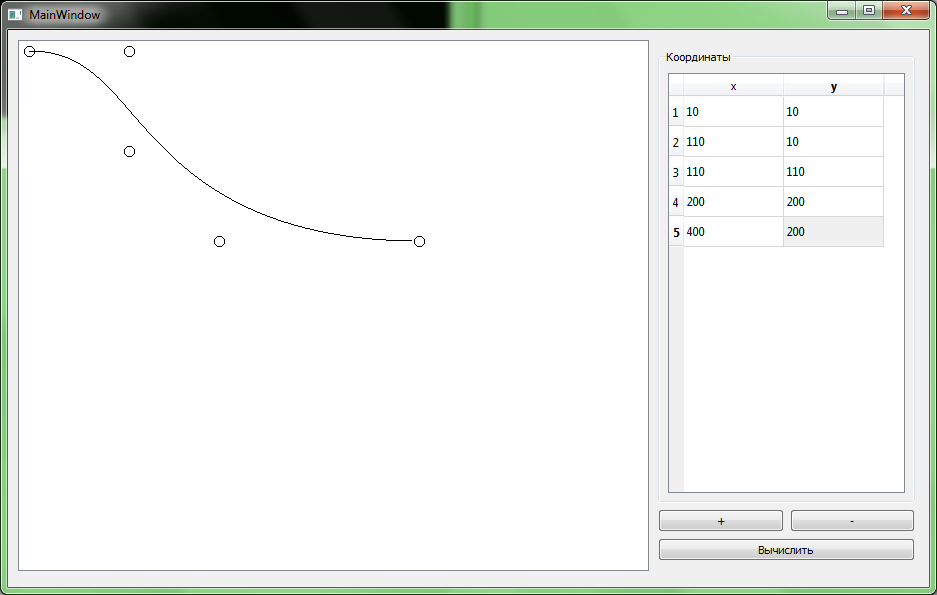
**Теоретические сведения:**



**Пример реализации алгоритма:**



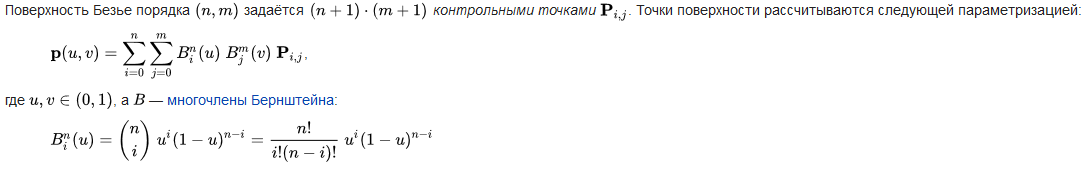
**Скриншот работы программы:**



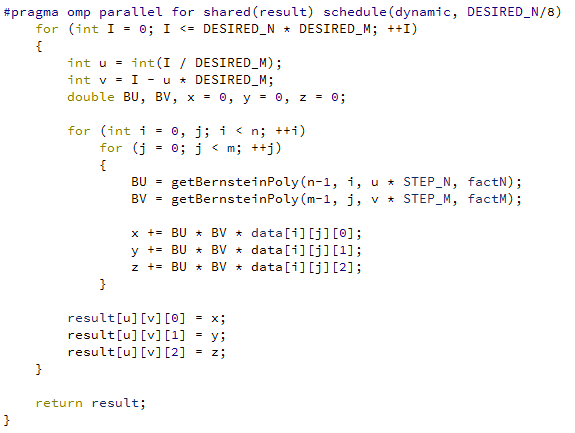
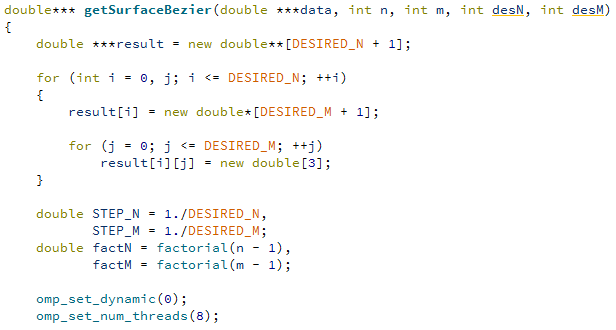
**3. Формирование произвольных поверхностей.**

**Задание:** сформировать поверхность Безье для различного задающего многогранника. Обеспечить поворот сформированной поверхности вокруг осей X и Y.

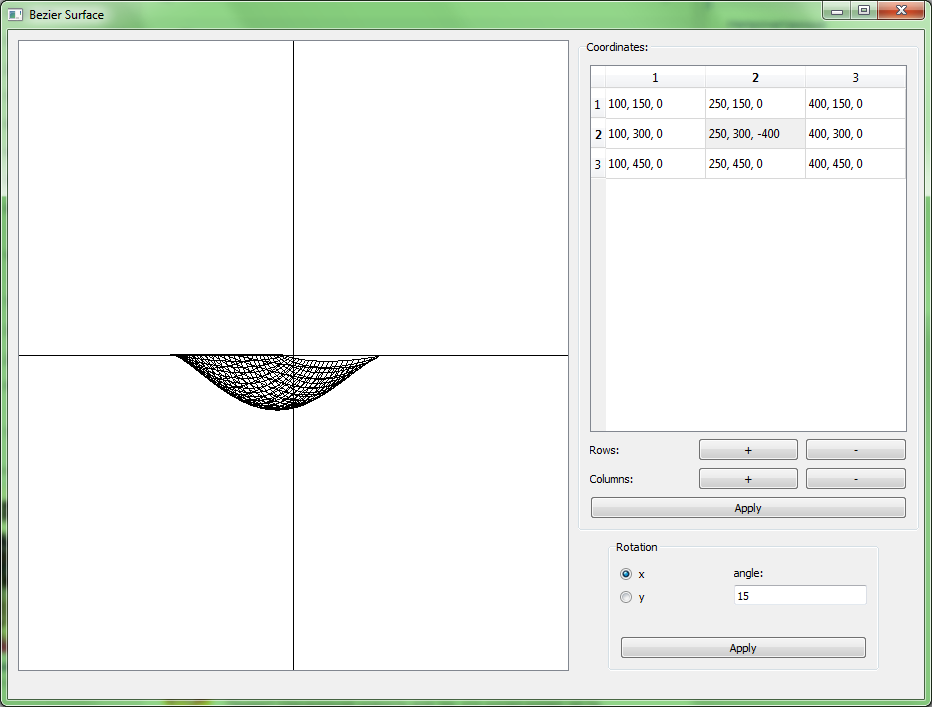
**Теоретические сведения:**



**Пример реализации алгоритма:**



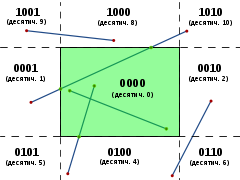
**Скриншот работы программы:**



**4. Исследование алгоритмов отсечения отрезков и многоугольников окнами различного вида**

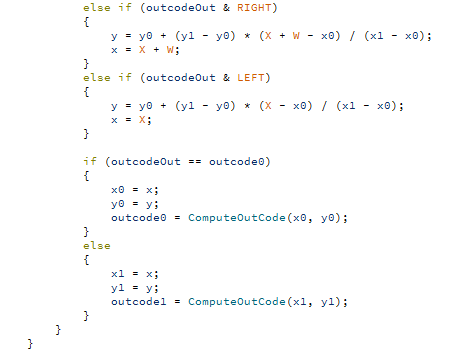
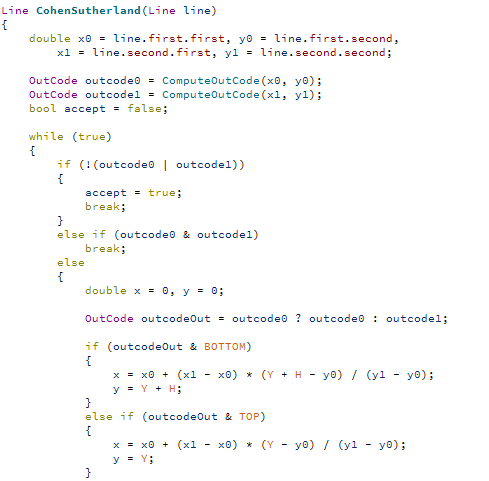
**Задание:** обеспечить реализацию алгоритма отсечения массива произвольных отрезков заданным прямоугольным окном с использование алгоритма Коэна-Сазерленда. Вначале следует вывести на экран сгенерированные отрезки полностью, а затем другим цветом или яркостью те, которые полностью или частично попадают в область окна.

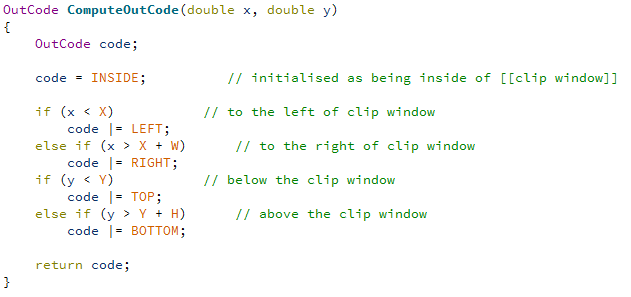
**Теоретические сведения:**

Алгоритм разделяет плоскость на 9 частей прямыми, которые образуют стороны прямоугольника. Каждой из 9 частей присваивается четырёхбитный код. Биты (от младшего до старшего) значат «левее», «правее», «ниже», «выше». Иными словами, у тех трёх частей плоскости, которые слева от прямоугольника, младший бит равен 1, и так далее.

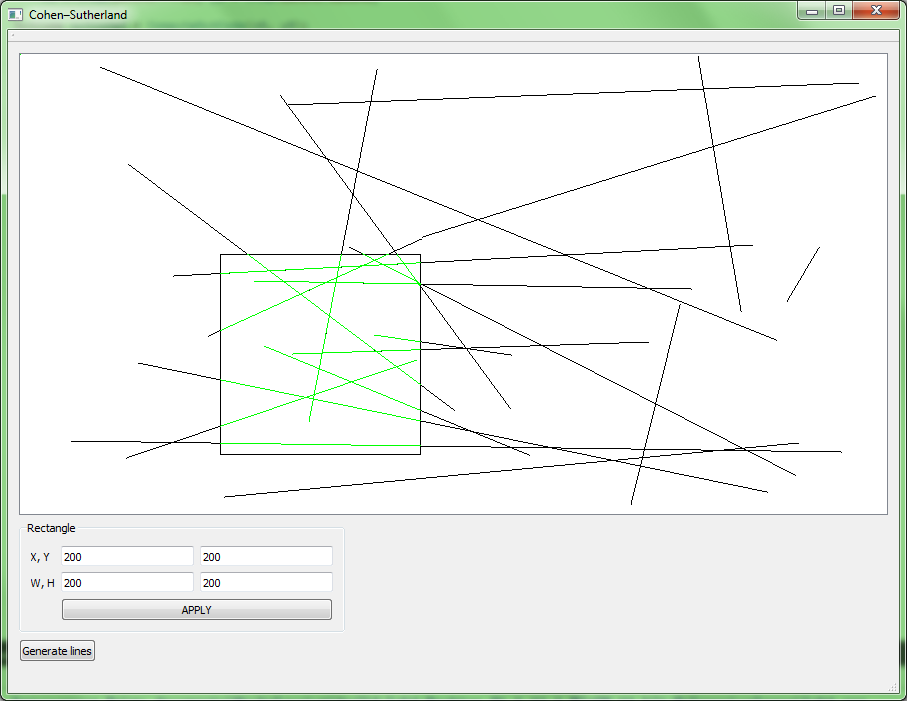
Алгоритм определяет код концов отрезка. Если оба кода равны нулю, то отрезок полностью находится в прямоугольнике. Если битовое И кодов не равно нулю, то отрезок не пересекает прямоугольник (так как это значит, что оба конца отрезка находятся с одной стороны прямоугольника). В прочих случаях, алгоритм выбирает конец отрезка (или один из концов), имеющий ненулевой код (то есть находящийся вне прямоугольника), находит ближайшую к нему точку пересечения отрезка с одной из прямых, образующих стороны прямоугольника, и использует эту точку пересечения как новый конец отрезка. Укороченный отрезок снова пропускается через алгоритм.

**Пример реализации алгоритма:**





**Скриншот работы программы:**



**5. Исследование алгоритмов выявления видимости сложных сцен**

**Задание:** обеспечить реализацию видимости совокупности произвольных многогранников на основе использования Z буфера.

**Теоретические сведения:**

Формальное описание алгоритма *z*-буфера таково:

1. Заполнить буфер кадра фоновым значением интенсивности или цвета.

2. Заполнить *z*-буфер минимальным значением *z*.

3. Преобразовать каждый многоугольник в растровую форму в произвольном порядке.

4. Для каждого *Пиксел*(*x,y*) в многоугольнике вычислить его глубину *z*(*x,y*).

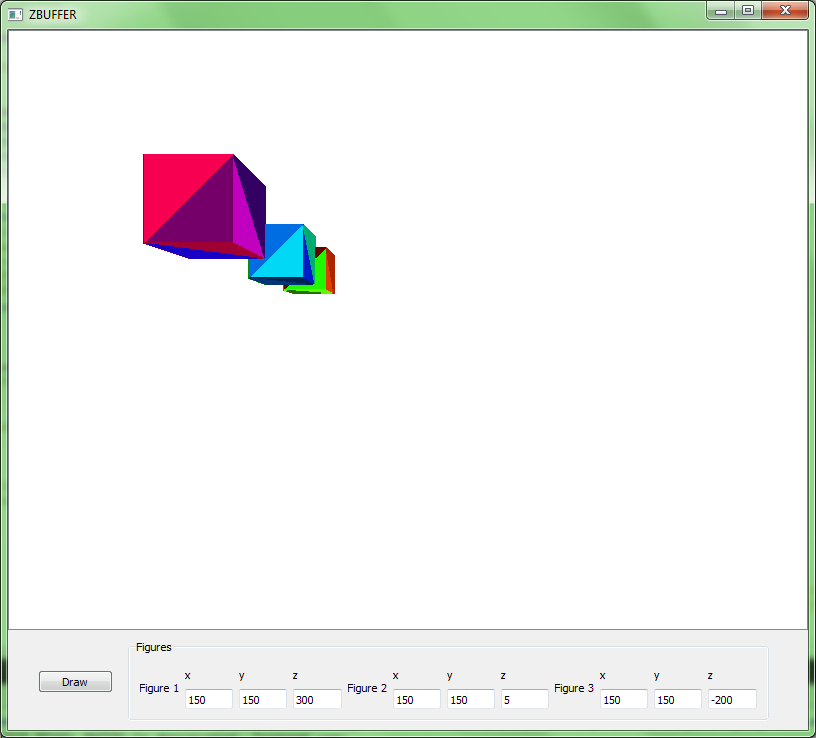
5. Сравнить глубину *z*(*х,у*) со значением *Zбуфер*(*х,у*), хранящимся в *z*-буфере в этой же позиции.

Если *z*(*х,у*) > *Zбуфер* (*х,у*), то записать атрибут этого многоугольника (интенсивность, цвет и т. п.) в буфер кадра и заменить *Zбуфер*(*х,у*) на *z*(*х,у*). В противном случае никаких действий не производить.

**Пример реализации алгоритма:**



**Скриншот работы программы:**



**6. Формирования реалистических изображений с использованием простых моделей освещения одним или двумя точечными источниками**

**Код шейдерной программы:**



**Скриншот работы программы:**

