Лабораторная работа №5

Алгоритмы удаления невидимых поверхностей

Цель работы: изучить алгоритмы удаления невидимых поверхностей и создать программу для визуализации объёмной трёхмерной модели с закрашенными гранями.

Порядок выполнения работы

Разработать алгоритм и составить программу для построения на экране трёхмерной модели с закрашенными гранями в соответствии с номером варианта лабораторной работы №4.

Требования к программе

1. В программе по центру окна должна отображаться только центральная проекция фигуры. Грани фигуры должны быть закрашены и отсортированы по глубине, т.е. пользователь должен видеть передние грани. Должна быть предусмотрена возможность задавать параметры центральной проекции: угол обзора или удалённость точки схода.
2. Пользователь должен иметь возможность поворачивать и перемещать фигуру вдоль оси с использованием мыши. Поворот фигуры лучше выполнять при обработке события mouseMoveEvent при зажатой кнопке мыши, перенос – при обработке собятия wheelEvent. Фигура должна поворачиваться «вслед» за мышью. Т.е., если мышь перемещается влево-вправо, то фигура должна поворачиваться вокруг вертикальной оси; если вверх-вниз, то вокруг горизонтальной оси.
3. В программе должна быть предусмотрена возможность ввода пользователем исходных данных (из правой колонки таблицы №1 лабораторной работы №4).

Краткие теоретические сведения

Для создания алгоритмов удаления невидимых поверхностей рассмотрим несколько вспомогательных алгоритмов.

Определение нахождения точки внутри выпуклого многоугольника.

(0,0)

*y*

*x*

*A*4

*A*3

*A*2

*A*1

*A*0

*O*

*a*4

*a*3

*a*2

*a*1

*a*0

Если точка *O* находится внутри выпуклого многоугольника, то для любого  
*i* (*i*=0,…,*n*-2) . **Векторное произведение векторов** на плоскости определяется как Если вектор находится правее вектора в порядке обхода по часовой стрелке, то векторное произведение векторов будет отрицательным, если левее – положительным. Для проверки нахождения произвольной точки внутри многоугольника необходимо для каждой вершины , за исключением последней, вычислить векторное произведение . Для последней вершины нужно находить произведение (на рисунке ). Если все векторные произведения будут иметь одинаковый знак, то точка *O* находится внутри многоугольника. Предполагается, что вершины многоугольника предварительно упорядочены по/против часовой стрелке.

Прежде чем создать алгоритм сортировки граней трехмерного объекта по глубине, рассмотрим более простой случай. Пусть в трёхмерном пространстве расположены два отрезка AB и CD, проекции которых на экранную (картинную) плоскость пересекаются. Необходимо определить, какой из них расположен ближе к экранной плоскости.

C’

A’

B’

D’

O

Проекции отрезков AB и CD на плоскости z=0

Обозначим проекции отрезков AB и CD на экранную плоскость как A’B’ и C’D’, а точку пересечения проекции отрезков на экранной плоскости как *O*. Проекции точек отличаются тем, что они имеют координату *z*, равную нулю. Необходимо найти точки пересечения прямой z\*, выходящей из точки O в направлении оси z с отрезками *AB* и *CD* в пространстве. Та точка пересечения, которая имеет меньшую координату z, расположена ближе к экранной плоскости. Обозначим точку пересечения прямой z\* и AB как .

Построим уравнение прямой, на которой лежит отрезок AB, и подставим в него координаты точки :

В данном уравнении неизвестная – . Для её нахождения необходимо сначала определить, какой знаменатель больше по модулю. Если , то находить решение системы:

Иначе, решать систему

Аналогично нужно определить удалённость точки пересечения отрезка *CD* и прямой z\* как решение системы:

Если , то отрезок *AB* перекрывает *CD* с точки зрения наблюдателя, т.е. *AB* расположен ближе, чем *CD*.

Рассмотрим более сложную задачу. Предположим, необходимо отсортировать по глубине два многоугольника. Возможны следующие случаи взаимного расположения проекций данных многоугольников:

1. Стороны многоугольников пересекаются.

В этом случае одна из пар сторон двух многоугольников обязательно пересекается. В общем случае нужно рассмотреть каждую пару отрезков (сторон) двух многоугольников и найти хотя бы одну пересекающуюся. Далее, используя вышеописанный алгоритм, не трудно определить какой отрезок (и соответственно многоугольник, к которому он принадлежит) расположен ближе к экранной плоскости.

1. Один из многоугольников находится внутри другого.

*A*2 (*x*2,*y*2,*z*2)

*A*1 (*x*1,*y*1,*z*1)

*A*4

*A*3

*E*

*A*0 (*x*0,*y*0,*z*0)

Для обнаружения этого случая нужно проверить каждую вершину первого многоугольника на расположение внутри второго, используя свойство векторного произведения векторов. В таком случае все вершины первого многоугольника находятся внутри второго, или наоборот. Порядок следования граней можно определить следующим образом:

1. Найти центр тяжести *E* первого многоугольника. Найти уравнение плоскости (например, по любым трём точкам), которой принадлежит второй многоугольник в виде .
2. Найти точку пересечения прямой с плоскостью  
   . Найти *z*, полученное из решения уравнения   
   . Это равносильно решению уравнения:
3. Если , то первый многоугольник расположен ближе к экранной плоскости, иначе – дальше.
4. Проекции многоугольников не пересекаются.

В данном случае порядок рисования многоугольников не важен.

Сортировка объектов по глубине

Самый простой алгоритм удаления невидимых поверхностей – сортировка граней объекта по глубине. Но этот алгоритм формирует изображение правильно только в том случае, если поверхность фигуры разбита на достаточное количество маленьких непересекающихся граней. Если это условие не выполнено, то в изображении возможны артефакты (нежелательные эффекты). Тогда пользователь увидит неправильный порядок сортировки граней. В этом случае просто необходимо каждую проблемную грань разбить на несколько граней.

Рассмотрим словесное описание данного алгоритма в совокупности с требуемыми аффинными преобразованиями:

1. Разместить фигуру вблизи начала координат. Например, пусть центр фигуры совпадает с началом координат.
2. Если необходимо, повернуть и масштабировать фигуру в начале координат.
3. Перенести фигуру вдоль оси z вглубь экрана.
4. Отсортировать грани фигуры по глубине в текущих координатах. Необходимо вычислить координату центра (тяжести) каждой грани и отсортировать весь массив граней по этому значению. После сортировки массив будет содержать грани в порядке от ближних к дальним (или наоборот). В дальнейшем будет достаточно нарисовать на экране все грани в порядке от дальних к ближним. Таким образом, ближние к наблюдателю грани будут нарисованы последними. Лучше использовать стандартную функцию сортировки со сложность  
   .
5. Выполнить операцию проектирования каждой грани фигуры на экранную плоскость и нарисовать её.