

# Специализированная Система Визуализации Некоторых Задач Оптимального Управления

А.В. Мошков, В.Ю. Пахотинских, В.О. Решетняк  
ИММ УрО РАН

[moshkov@list.ru](mailto:moshkov@list.ru), [reshetnyakv@yandex.ru](mailto:reshetnyakv@yandex.ru).

## Abstract

В работе описывается специализированная система визуализации для одной из задач оптимального управления. Основное внимание уделяется методам обработки и хранения больших объемов данных с применением фильтрации по средством конвертирования в различное представление (воксельное и полигональное) и его последующей визуализации. При этом сохраняется внутренняя и внешняя структура объектов. Отдельно рассматривается возможность задания степени детализации для обеспечения гибкости решения задачи.

*Keywords: оптимальное управление, трехмерная визуализация, воксельная графика, полигональная графика, триангуляция, октавное дерево.*

## 1. ВВЕДЕНИЕ

Поиск областей достижимости в задачах оптимального управления послужил источником нашей постановки. Математику необходимо изучать общий вид и внутреннюю структуру области достижимости, причем огромный объем полученного расчетного материала требует использования средств трехмерной визуализации.

Ряд причин, связанных с реализацией алгоритма, привел к тому, что вычисленные данные хранятся в формате \*.bmp, при этом входные данные составляют порядка миллионов, а то и миллиардов точек. Что и породило комплекс проблем, решенных нами при разработке системы.

В результате создан комплекс программ, позволяющий провести фильтрацию данных не искажающую ни внешнюю и ни внутреннюю структуру объекта. И позволяет отобразить как процесс построения, так и сам объект в разных ракурсах.

## 2. ОБЩАЯ СТРУКТУРА ПРОГРАММНОГО КОМПЛЕКСА

Данный программный комплекс состоит из набора утилит для работы с огромным облаком точек и его последующей визуализации. В разработке использованы технологии OpenGL и DirectX.

Стадии конвейера:

- Обработчик bitmap файлов
- Вычислитель освещенности
- Конвертер в воксельный формат и создание структур хранения сцены
- Сглаживатель воксельных объектов
- Конвертер в полигональный формат (триангулятор)

## 3. ВХОДНЫЕ ДАННЫЕ

На вход поступает набор точек в трехмерном пространстве, заданный в виде плоских срезов по оси Z, и хранимые в файлах формата \*.bmp.

Преимущества хранения в \*.bmp: если информацию о точке хранить в виде координат (x,y,z) типа double, то информация об одной точке будет занимать 24 байта. Мы используем монохромные bitmap, поэтому объем информации о точке можно считать приблизительно равным 1 биту (если отбросить заголовок \*.bmp файла и пустое пространство в файле). Таким образом с использованием bitmap файлов объем хранимой информации уменьшается в  $24 \cdot 8 = 192$  раза. На самом деле это достаточно грубая оценка, так как заполняемость bitmap файла точками зависит от конкретной системы.

Если использовать сжатие файлов (например, встроенное сжатие файлов NTFS), то потребуется еще меньше дискового пространства для сохранения информации о состоянии системы. Следует отметить, что степень сжатия монохромных bitmap файлов очень высокая, в отличие от бинарных файлов.

## 4. ПРЕДОБРАБОТКА ДАННЫХ

### 4.1. Сортировка и нормирование

Полученный файл подвергается обработке. Последовательно считываются все bitmap файлы и «на лету» сортируются в нужной последовательности и нормируются. Получаем файл с данными, подготовленными для дальнейшей обработки, содержащий тройки координат (x,y,z).

### 4.2. Вычисление освещенности

Для улучшения восприятия трехмерности представления сцены используется освещение.

Освещенность рассчитывается методом интерполяции векторов нормалей. Вектор нормали в точке является вектором нормали аппроксимирующей плоскости, соседних точек, лежащих на определенном расстоянии от заданной точки. Расстояние является переменной величиной. В зависимости от изменения величины этой переменной меняется степень освещенности.

Полученный файл побочно считывается в оперативную память с использованием технологии FileMapping, после чего в этом блоке рассчитывается освещенность, и весь блок вместе с полученными значениями освещенности записывается в бинарный файл. В конце данной операции получаем сжатый файл, в котором хранятся четверки чисел (x, y, z координаты и цвет).



