

# Лекция №10

## Приёмы оптимизации сцены Обработка изображений

Рябинин Константин Валентинович

e-mail: [icosaeder@ya.ru](mailto:icosaeder@ya.ru)

jabber: [icosaeder@jabber.ru](mailto:icosaeder@jabber.ru)

Пермь, 2011

**Под оптимизацией процесса рендеринга понимается комплекс мер, которые можно предпринять, чтобы увеличить производительность графической программы**

**Различают несколько базовых подходов:**

- Разумная экономия на разрешении графических данных (трёхмерных моделей и текстур)**
- Использование встроенных в конкретную систему визуализации средств оптимизации (например, использование буферных объектов в OpenGL)**
- Активное искусственное упрощение сцены без снижения качества результирующей картинки**
- Оптимизация расчётов (отказ от точных вычислений в пользу быстрых аппроксимаций)**
- . . .**

**Под упрощением сцены понимается сокращение количества выводимых объектов и их геометрической сложности**

**Существует большое количество алгоритмов такого упрощения:**

- **Z-буферизация**
- **Отсечение задних граней**
- **Отсечение по усечённой пирамиде видимости**
- **Отсечение заслонённых объектов**
- **Портальный рендеринг**
- **Уровни детализации**
- **...**

- Данный алгоритм был подробно рассмотрен ранее
- Идея – в процессе рендеринга выполнять отсечение частей объектов, закрытых частями более близко расположенных объектов
- Повысить эффективность z-буферизации способен **алгоритм художника** – сортировка объектов по удалённости и вывод их, начиная с самых ближних
- Однако алгоритм художника не применим в том случае, если для объектов используется альфа-смешивание. В этом случае, для получения корректного результата необходимы прямо обратные действия (вывод объектов, начиная с самого дальнего)

**Отсечение задних граней** (*backface culling*) – это алгоритм, при котором

- Для каждого полигона определяется, внешней или внутренней стороной он повёрнут к наблюдателю (определение производится на основе порядка обхода вершин)
- В зависимости от результата определения принимается решение, выводить данный полигон, или нет
- Алгоритм позволяет сэкономить на задних гранях объекта, для которых делается предположение, что они закрыты передними
- Алгоритм применим только для «замкнутых» объектов (полностью ограничивающих какую-то часть пространства)
- Отсечение граней по их ориентации относительно наблюдателя поддерживается библиотеками визуализации. Например, в OpenGL его можно включить вызовами  
`glEnable(GL_CULL_FACE);`  
`glCullFace(GL_BACK); // либо glCullFace(GL_FRONT);`

**Отсечение по усечённой пирамиде видимости**  
(*frustum culling*) – это алгоритм, при котором

- Для каждого объекта определяется, попадает ли он в поле зрения камеры (в роли поля зрения камеры выступает, в общем случае, усечённая пирамида видимости)
- Объект отправляется на графический конвейер только в том случае, если он находится в поле зрения камеры
- Для граничных случаев (часть объекта в поле зрения камеры, а часть – за пределами) существуют различные подходы, но чаще всего, чтобы не тратить время на разбиение объекта на множество подобъектов и вычленение из них видимых, объект полагается видимым и отрисовывается
- Определение попадания в поле зрения осуществляется чисто геометрически, на основе уравнений плоскостей, составляющих усечённую пирамиду видимости
- Геометрический тест позволяет определить, находится ли точка в поле зрения. Производить этот тест для всех вершин объекта неэффективно, поэтому используются т. н. **ограничивающие объёмы**

**Отсечение по усечённой пирамиде видимости**  
(*frustum culling*) – это алгоритм, при котором

- **Ограничивающий объём** – это геометрическое тело простой формы, описанное вокруг объекта
  - Чаще всего в качестве такого тела используется **параллелепипед** (*bounding box*) или **сфера** (*bounding sphere*)
  - Использование параллелепипеда является более гибким, так как, обладая тривиальной формой, он способен, однако, достаточно точно ограничить произвольный объект
  - Тест видимости производится для ограничивающего объёма, а не для самого объекта
- Помимо этого, ограничивающие объёмы могут быть использованы для определения столкновений объектов, при моделировании физики твёрдого тела

**Отсечение заслонённых объектов** (*occlusion culling*) – это алгоритм, при котором

- Заранее определяются объекты, полностью закрытые другими объектами
- Невидимые объекты не отправляются на графический конвейер
- Отличие от z-буферизации состоит в том, что невидимые объекты *не отправляются на конвейер*
- Поиск заслонённых объектов – сам по себе довольно сложная задача. Поэтому использование этого алгоритма имеет смысл лишь
  - На сценах с большим количеством объектов разного размера
  - На сценах, моделирующих помещения (в которых, например, присутствуют стены и перегородки)



**Портальный рендеринг** (*portal rendering*) – это алгоритм визуализации закрытых пространств, при котором

- Для сцены, представляющей собой цепочку комнат, соединённых дверными и оконными проёмами (**порталами**), определяется видимая в данный момент площадь
- На конвейер отправляются только те объекты, которые принадлежат этой площади
- Для эффективного поиска видимых областей и объектов, принадлежащих этим областям, используется деление пространства (*space partitioning*)
- Деление пространства бывает двоичным (*BSP*), квадратичным (*quadtree*) и октарным (*octree*)
- Существует целый ряд алгоритмов деления пространства
- Результат деления сохраняется в древовидную структуру (бинарные, квадратичные и октарные деревья соответственно)
- Деление пространства используется для быстрого поиска объектов и определения их взаимного расположения
- Однако на динамических сценах, когда положение объектов постоянно изменяется, вместо прироста производительности можно получить её падение, так как деревья придётся всё время перестраивать

**Уровни детализации** (*LOD – level of detail*) – это изменение детализации объектов в зависимости от их удалённости

- Частным случаем LOD является мип-мэппинг текстур
- Идея состоит в том, что для удалённых объектов снижается детализация их представления
- Различные уровни детализации чаще всего генерируются автоматически на основе оригинала, разрешение которого берётся за максимальное
- Однако иногда разные уровни детализации подготавливаются заранее и загружаются из файлов
- Проблемой является переключение между уровнями. Если в случае текстур можно использовать интерполяцию цвета для сглаживания «скачка» с одного уровня на другой, то в случае трёхмерных моделей простая интерполяция невозможна
- Для трёхмерных моделей могут быть использованы алгоритмы **морфинга** (плавного перехода между формами), однако чаще всего используется скачкообразное изменение
- В качестве крайней меры снижения детализации трёхмерных объектов могут быть использованы импостеры, которые были рассмотрены ранее

**Фильтрация изображения** – это процесс изменения исходного изображения при помощи некоторой функции (фильтра)

- Вообще говоря, фильтром может называться любая функция, изменяющая изображение
- Фильтр принимает на вход исходное изображение и, возможно, некоторые дополнительные данные, как связанные с изображением, так и не связанные с ним
- Результатом применения фильтра является другое изображение, вообще говоря, несводимое к исходному
- Фильтрация применяется для
  - Повышения качества изображения
  - Достижения специального визуального эффекта
  - Адаптации изображения к условиям конкретной задачи (например, при распознавании образа)

**Конволюция** – это процесс замены точки изображения на средневзвешенное (с различными весовыми коэффициентами) значение его малой окрестности

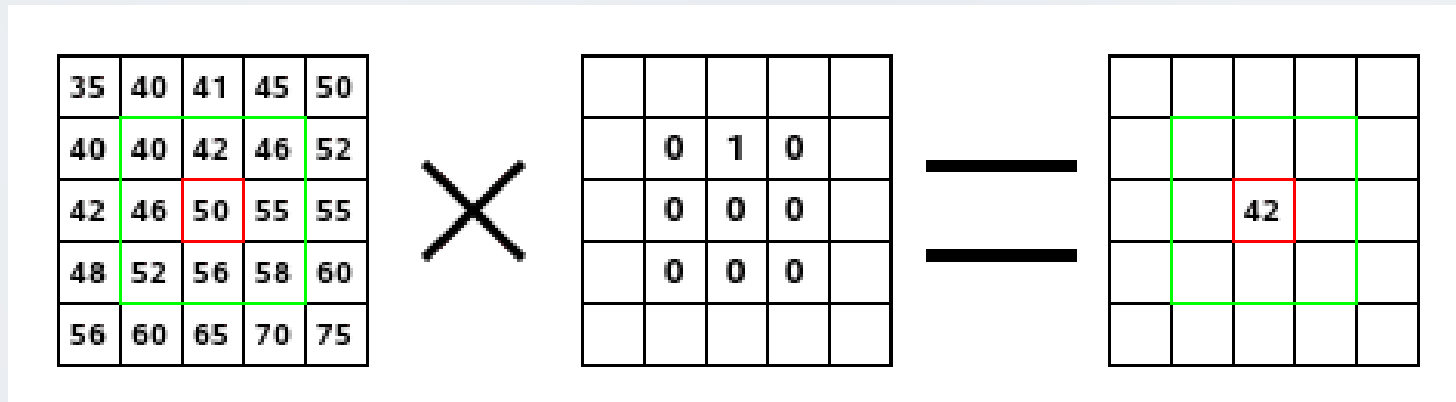
**Матрица конволюции** – это матрица, задающая правило вычисления средневзвешенного значения

**Общий алгоритм применения матрицы конволюции:**

- Пусть есть матрица  $m \times n$  (чаще всего  $m = n$ )
- Апертура («окошко») размером  $m \times n$  пробегает своей *центральной точкой* по всем точкам изображения
- Цвет каждой точки умножается на соответствующий коэффициент, взятый из матрицы
- Полученные значения складываются и делятся на сумму всех коэффициентов (нормируются). Возможно, к нормированной величине прибавляется некоторая константа
- Полученное значение присваивается точке изображения, которая соответствует центральной точке апертуры

- **Применение конволюции – это наиболее популярный алгоритм фильтрации изображений, так как, обладая простой реализацией, этот способ покрывает огромное количество самых разнообразных эффектов, от коррекции резкости до выделения контуров и рельефа**
- **Достоинством алгоритма является линейное время фильтрации**
- **На основе конволюции реализуются и более сложные – многопроходные – фильтры, представляющие собой дифференциальные операторы (дифференциал аппроксимируется свёрткой, получаемой с помощью матрицы конволюции)**

- Чаще всего матрица конволюции является квадратной и имеет размерность 3x3 или 5x5



- Во многих графических редакторах (например, в GIMP) матрица конволюции, элементы которой может задавать пользователь, выделена в отдельный фильтр
- Для вывода весовых коэффициентов используется аппарат высшей математики (например, в процессе работы с дифференциалами цвета)

## Примеры:

### ● Увеличение контраста

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$



### ● Равномерное размытие

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$



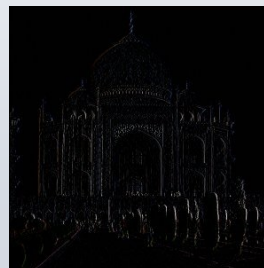
### ● Размытие по Гауссу

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$



### ● Усиление карёв

$$\begin{pmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

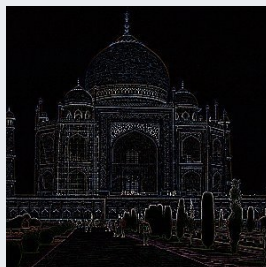




## Примеры:

### ● Выделение контура

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$



### ● Придание рельефа

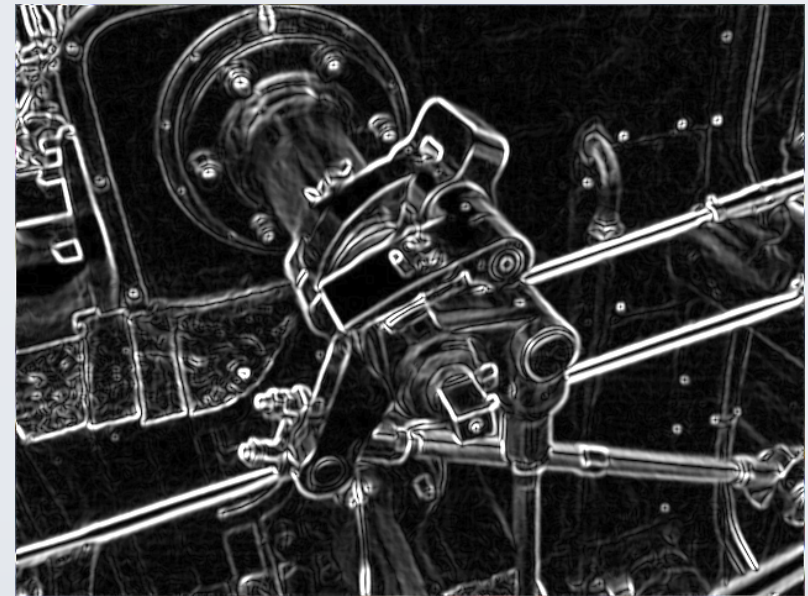
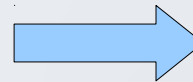
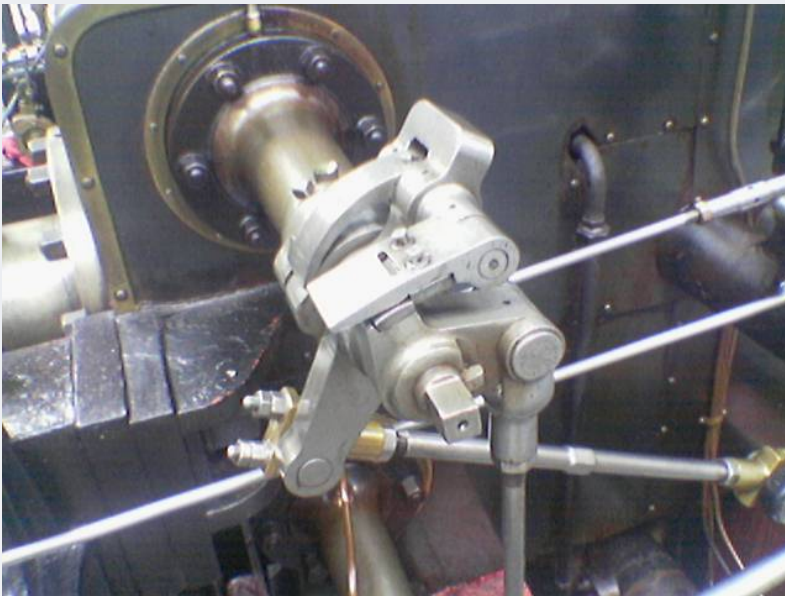
$$\begin{pmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$





Более сложный пример – **дифференциальный оператор Собела**

Фильтр Собела визуализирует градиент яркости, в результате чего на изображении подсвечиваются границы



**Фильтр Собела требует уже двух проходов с разными матрицами конволюции (для вычисления X и Y координат градиента в каждой точке):**

$$G_x = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \quad G_y = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

$$I' = \sqrt{(G_x[I])^2 + (G_y[I])^2}, \text{ где}$$

*I* – исходное изображение

*I'* – результирующее изображение

*G[I]* – конволюция изображения

*Операции возведения в степень выполняются покомпонентно*