Методы построения нефотореалистичных изображений и перспективы их использования в комплексных системах

М. Самохвалов

1. Нефотореалистичная графика

1.1. Введение

В настоящее время компьютерная графика прочно вошла в нашу жизнь. «Игрушечная история» (“Toy Story”), «Титаник», «Из жизни насекомых» (“A Bug’s Life”), различная реклама, журналы... Многое из того, что мы видим в экране телевизора, на улицах, в Интернете сделано при помощи программ, которые предназначены для создания таких изображений и видео, которые сложно или невозможно получить при помощи фотографии, видеосъемки или каких-либо других методов.

Эти инструменты включают в себя модули, которые предназначены как для моделирования и анимации 3-мерных объектов, так и для их визуализации. Процесс перевода 3-мерных сцен в изображения называется рендерингом. На данный момент разработаны методы, позволяющие получить качество, сравнимое с фотографией. Они включают в себя наложение реалистичных материалов на 3-мерные модели, построение естественного освещения, физические модели сред и т. д.

Теория, которая лежит в основе этих методов, основана в основном на физике, но в последнее время стали использоваться методы, которые имеют другую основу. Эти методы используются для построения изображений, которые помогают выделить детали, которые не видны при использовании реалистичных методов. Например, изображения, полученные при томографии, с большим трудом поддаются визуальному анализу без дополнительной обработки. Для выделения нужной информации нужно обработать изображение так, чтобы существенные детали были выделены. Также иногда требуется, чтобы в изображение были добавлены элементы, которые изначально не присутствовали для придания большей выразительности картинке. Это относится, например, к таким приложениям, как мультипликация и кино. Графика, использующая эти методы, в силу своей специфичности называют нефотореалистичной.

Таблица 1

	Фотореалистичная графика	**Нефотореалистичная графика**
Основа используемых алгоритмов	Симуляция физических процессов	Как ориентированные на стилизацию, так и специализированные методы.
Качество получаемой картинки	Фотографическое	Стилизованное
Уровень детализации	Сложные методы для получения точной детализации, большое количество дополнительной информации.	Адаптивная детализация
Полнота изображения	Полное, с мелкими деталями	Неполное, детали могут отсутствовать
Точность изображения	Точно	Приблизительно

Область применений нефотореалистичной графики довольно широка. Обработка изображений и трехмерных данных нужна в медицине (описано выше), строительстве (для выделения общих контуров здания, придание чертежу стиля «рисованного от руки» для заказчика и т. д.), технической иллюстрации, в других областях, где нужно схематично показать данные.

Основные моменты различий между фото и нефотореалистичной графикой показаны в табл. 1.

Программы трехмерной анимации, так же как и аппаратные платформы, на которых они выполняются, стали более производительными и удобными в эксплуатации, а также доступными по цене. В итоге большинство студий компьютерной графики стало переходить на трехмерные технологии. При этом стоимость производства снижается в отдельных случаях на несколько порядков. Но, к сожалению, при этом она остается высокой, потому что часто приходится затрачивать много усилий на совмещение разных технологий визуализации.

1.2. Нефотореалистичная графика

Решений построения фотореалистичной графики уже достаточно много. Но на данный момент не существует общего решения проблемы построения на компьютере изображений нефотореалистичных. Студии применяют свои решения для создания рисованных мультфильмов и нефотореалистичной графики, которые приспособлены под их конкретные проекты. К таким решениям можно отнести стилизованный рисунок, двух-

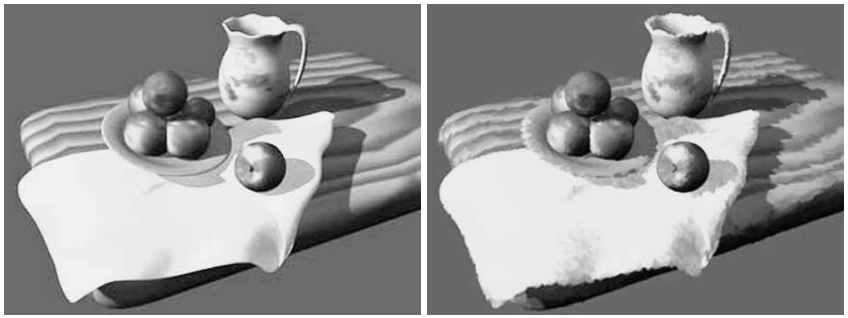


Рис. 1. Отличительные особенности (слева — фотореализм, справа — нефотореалистичное изображение painterly rendering)

мерную мультипликацию, имитацию холста и т. д. Сложность построения таких изображений состоит в том, что при применении фотореалистичных технологий полученная картинка не содержит информации, которая присутствует в рисунках и анимации, сделанных художниками, например такой, как линии контура, толщину мазков на бумаге, фактуру холста и т. д.

Есть несколько наиболее хорошо известных методов для построения нефотореалистичных изображений, которые описаны в литературе. Мы рассмотрим их, а также более или менее распространенные средства подготовки трехмерных сцен и анимации для полиграфии, публикации в Web, кино и видео.

В последнее время стало много внимания уделяться именно «мультипликационному» рендерингу. Эта технология сейчас используется многими компаниями для упрощения процесса производства мультфильмов.

На данный момент разработкой методов для обработки и получения изображений также занимаются такие организации, как:

- Brown University,
- University of Utah,
- INRIA (.fr),
- New York University,
- Princeton University,
- University of Wales Swansea.

А также ежегодно собирается конференция, посвященная нефотореалистичной графике NPAR (www.npar.org).

Также в рамках конференции SIGGRAPH проводятся семинары по этой теме.

2. Обзор методов нефотореалистичного рендеринга

Методов построения нефотореалистичных изображений очень много, но условно их можно поделить на две части: построение контура картинки

и заливка контура. Также можно их поделать на две независимые группы по построению контура: методы, основанные на обработке изображений и методы, в которых контур получается в результате анализа 3-мерной геометрии.

Также есть области этой дисциплины, в которых применяется только обработка изображений и разрабатываются двухмерные фильтры для построения изображений из видеопотока и/или из двухмерных изображений без построения контура.

2.1. Терминология и основы

2.1.1. Классификация контура

В литературе, посвященной нефотореалистичной графике, очень много внимания уделено именно контуру. Ниже приведена классификация граней, из которых состоит контур при представлении модели в виде некого множества полигонов.

1. Силуэт. Эти грани находятся на стыках 2 полигонов, один из которых повернут лицевой стороной к камере, а другой — нелицевой. Математически это выражается в том, что векторное произведение вектора камеры на нормали полигонов имеет разные знаки.
2. Бордюр. Эта грань имеет граничным к себе только один полигон.
3. Складка. Это грани между полигонами, векторное произведение которых с вектором камеры имеет один и тот же знак. Однако при этом угол между полигонами имеет значение больше некого граничного, вследствие чего по этим граням можно построить контур.

2.2. Анализ изображения (2-мерные методы)

2.2.1. Построение контура

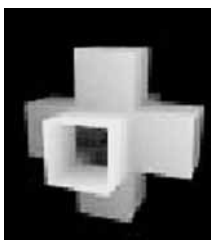


Рис. 2. Карта глубины

Один из наиболее распространенных методов получения контура — это использование карт глубины и нормалей [2]. Этот метод основан на применении фильтров семейства “edge detection” к картам глубин и нормалей.

Карта глубины (см. рис. 2) представляет собой изображение, в котором интенсивность пикселя пропорциональна глубине точке сцены, спроецированной в него. Идея метода основана на предположении, что изменение глубины между пикселами обычно гораздо меньше в пределах одного объекта, чем между несколькими объектами. Алгоритмы таких фильтров можно найти в литературе (например, метод «Собеля»). Большинство графических пакетов предоставляют возможность получить карту глубин для трехмерной сцены.

Есть несколько проблем, связанных с применением только карт глубины. Это невозможность отличить один объект от другого, если они находятся на одинаковом расстоянии от проекционной плоскости, а также

невозможность обнаружения складок на поверхности. Эти недостатки можно свести к минимуму, используя нормали к поверхности. Это, как правило, делается путем использования карт нормалей.

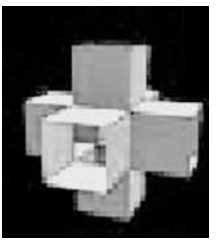


Рис. 3. Карта нормалей

Карта нормалей (см. рис. 3) — это изображение, которое представляет нормаль к поверхности в каждой точке изображения. Значения RGB компонент цвета точки на карте нормалей представляют соответственно XYZ нормали поверхности в этой точке.

Мы также можем использовать наш фильтр для получения граней, составляющие контур для нашей сцены по этой карте. Эти грани соответствуют изменениям ориентации поверхности и могут быть скомбинированы с гранями, полученными из карты глубины.

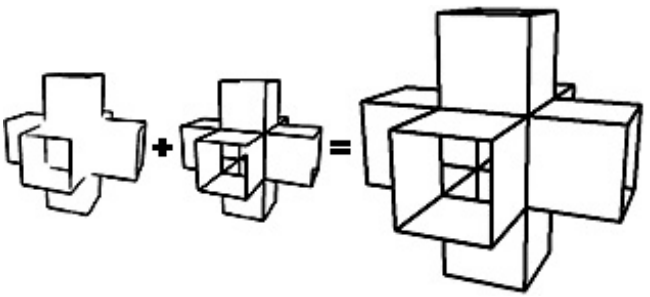


Рис. 4. Контур, полученный из карты глубины, + карты нормалей = результирующее изображение

На рис. 4 показан результат получения контура из этих 2 карт. На нем видно, что информации, полученной только с одной из них, недостаточно для получения качественного изображения.

2.2.2. Закрашивание контура

Процедура закрашивания контура в двухмерном случае принципиально не отличается от обычного закрашивания (текстурирования) трехмерной модели. Однако есть несколько методов, которые отталкиваются от полученных изображений и не используют трехмерной информации. Как правило, это различные комбинации двухмерных фильтров, которые после получения контура применяются к исходному изображению для получения нефотореалистичного качества (пример показан на рис. 5).

Например, для получения имитации рисунка можно к фотографии применить следующую методику.

Возьмем обычное изображение. Далее мы будем удалять части изображения до получения нефотореалистичного эффекта. Удалим границы

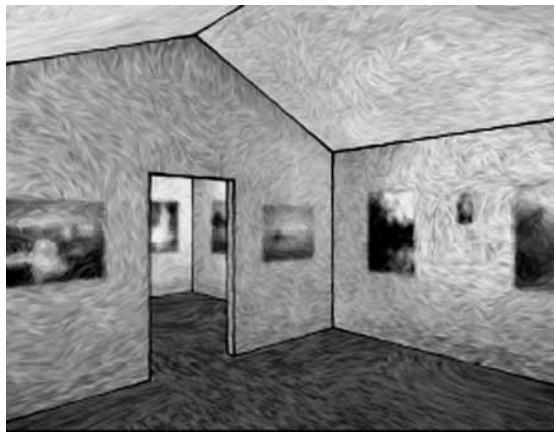


Рис. 5. Рисунок взят из работы [9], посвященной построению нефотореалистичного окружения

изображения. Удаляя границы, мы добиваемся того, что картинка становится подобна тому, как будто их на самом деле нарисовали на холсте. Теперь пришло время специального фильтра — текстуры холста.

Возьмем, например холст «фактура» (в программе для рисования Photoshop это встроенный фильтр и можно самому попробовать повторить шаги). Наложение фактуры происходит автоматически. Технологически это двухмерный фильтр, который накладывает полупрозрачную текстуру на картинку. При этом при наложении можно использовать маску для тех мест, где фактура нежелательна.

Далее можно придать законченность нашему изображению. Что отличает одного художника от других — способ, которым он рисует. Поэтому хотя наш рисунок и выглядит просто замечательно, но все же не так, будто бы его кто-то рисовал на самом деле. Мы можем имитировать темные мазки, как будто мы на самом деле рисуем эту картину и хотим выделить какие-то ее части.

В автоматическом режиме можно имитировать это, пользуясь фильтром для выделения контура, чтобы определить наиболее контрастные части изображения и «обвести» этот контур кистью.

В результате мы получим изображение на рис. 6.

Конечно, эта методика работает гораздо лучше на «не столь реалистичных» изображениях, так как большинство художников не могут достичь такого результата, который мы получаем в данном случае при помощи таких программ, как Lightscape или Maya. Поэтому любое приближение к нефотореалистичному изображению оказывается очень полезным.

Мы видим, что незначительная по трудоемкости обработка изображения привела его к нефотореалистичному виду. К сожалению, анимировать

Рис. 6. Слева — исходное изображение. Справа — полученное

серию таких изображений практически нереально, так как будут появляться артефакты, связанные со случайностью накладываемых «мазков», а также с «дребезгом» цветов после наложения фильтров. Этот метод подходит только для создания статических изображений.

2.3. Анализ 3-мерной геометрии

2.3.1. Закрашивание контура

Закрашивание контура при использовании трехмерных данных довольно просто. Обычно в production для этого используются так называемые toon-шейдера. Математика этих шейдеров не сложная и может быть реализована даже в реальном времени (не стоит забывать, что это реализуемо в достаточно простых случаях). В Интернете есть много примеров на эту тему. В данном обзоре мы опишем технологию закрашивания, использованную в [8], а также в другом разделе опишем построение такого шейдера для программы трехмерного моделирования, применяемой в производстве мультипликации компании Alias, называемой Maya.

«Мультипликационные» персонажи представляются двухмерными. И художники сознательно уменьшают количество видимых деталей при их рисовании. Они обычно используют жестко заданные цвета при рисовании персонажей, вместо использования технологий с реальным посчитанным освещением. Различие между этими технологиями очень простое: при использовании жестких цветов появляется четкая грань, отделяющая освещенную сторону персонажа, от той, которая попадает в тень. Эта технология заливки называется *hard shading*.

Соответствующая технология, применяемая в [8] для закрашивания, основана на вычислениях обычного диффузного освещения (см. рис. 7).

Выражение для этого приведено ниже:

$$C_l = a_g \times a_m + a_l \times a_m + \max\{\overline{L} \cdot \overline{n}, 0\} \times d_l \times d_m,$$

где C — цвет вертекса, a — коэффициент амбиентного освещения (индексы: l — источника света, m — материала), d — коэффициент

Рис. 7. Результирующее изображение

диффузного освещения, $\overline{L}$ — вектор из источника света на вертекс, $\overline{n}$ — нормаль к поверхности в данном вертексе.

Математика для мультяшного затенения приблизительно такая же. Вместо вычисления цветов для каждого вертекса мы будем использовать карту текстуры с минимальным количеством цветов. В большинстве случаев достаточно только двух: один для освещенного цвета и один для затененного.

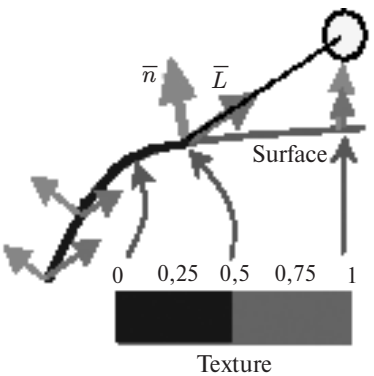


Figure 2: Generation of texture cooredinates from $\overline{L} \cdot \overline{n}$. In this case, the shadow boundary occurs at the point where $\overline{L} \cdot \overline{n}$ equals 0,5.

Рис. 8. Генерация текстурных координат. Ниже показана одномерная текстура

Основной цвет текстурной карты вычисляется заменой скалярного произведения в нашем выражении значением 1 (которое эквивалентно углу в 0° и означает, что источник света под прямым углом светит на поверхность). Цвет тени карты текстуры вычисляется заменой скалярного произведения значением 0 (которое означает, что поверхность освещена только амбиентным источником света). Результирующая текстура являет-

ся одномерной, вычисляется один раз на каждый материал и сохраняется в процессе просчета результирующего изображения.

Граница между освещенными и затененными местами на модели зависят от косинуса угла между вектором на источник света и нормалью. Мы вычисляем на каждом кадре $\max\{\overline{L}\cdot\overline{N},0\}$ для каждого вертекса и используем эти повертексные значения как текстурные координаты для нашей карты текстуры. На рисунке выше показано, как позиция источника света и направление нормали используются как индексы для одномерной карты текстуры. Заметим, что в этом методе мы вычисляем скалярное произведение для каждого вертекса в каждом кадре. На данный момент современные компьютеры могут вычислять освещение на таких моделях на центральном процессоре, но можно также использовать и аппаратную реализацию, используя новые видеокарты. В этом случае заливка моделей может быть выполнена в реальном времени. На данный момент выпущено даже несколько игр, использующих полностью или частично подобные технологии.

2.3.2. Закрашивание (дополнение)

В технологии, приведенной в [8] расширен метод для закраски поверхностей. Вместо выбора текселей в одномерной текстуре алгоритм может использовать также изображения с высокой детализацией для получения другого нефотореалистичного эффекта — имитация карандаша. Для этого применяется метод выбора подходящей текстуры для плотности. Регион изображения, получающий меньше света, имеет в случае такой закраски текстуру с более высокой плотностью штрихов. В регионах, где освещенность еще меньше комбинируются горизонтальные и вертикальные штрихи. Также используется фактура бумаги для более полного «реализма», которая комбинируется с текстурой штрихов.

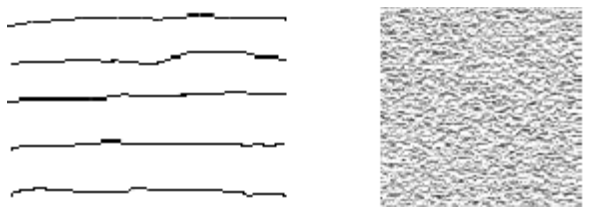


Рис. 9. Текстура штрихов и текстура бумаги

Если модель отскалирована, то текстура бумаги будет иметь такой же коэффициент масштабирования (см. рис. 10).

Этот метод показывает, как простыми средствами можно добиться высокого качества картинки. Однако при анимации таких сцен технология накладывания текстуры карандаша имеет недостатки. В анимации появляется эффект мерцания, т. е. наша модель будет «плавать» поверх

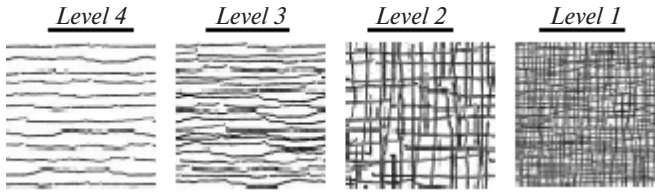


Рис. 10. Текстуры для нескольких уровней закраски

текстуры. При таких обстоятельствах может помочь идея «подгонки» текстурных координат при препроцессинге кадра, но это не рассматривается в работе [8].

2.3.3. Шейдера

Одним из наиболее простых методов нефотореалистичного рендеринга считаются так называемые «шейдера». Это небольшие специализированные программы, написанные, как правило, на языках высокого уровня, предназначенные для создания материалов, которые впоследствии накладываются на трехмерные поверхности в системах моделирования. Шейдера для удобства конечных пользователей часто изображаются как сеть из нескольких узлов, в которых происходят вычисления. Но это не является единственным способом представления материала поверхностей.

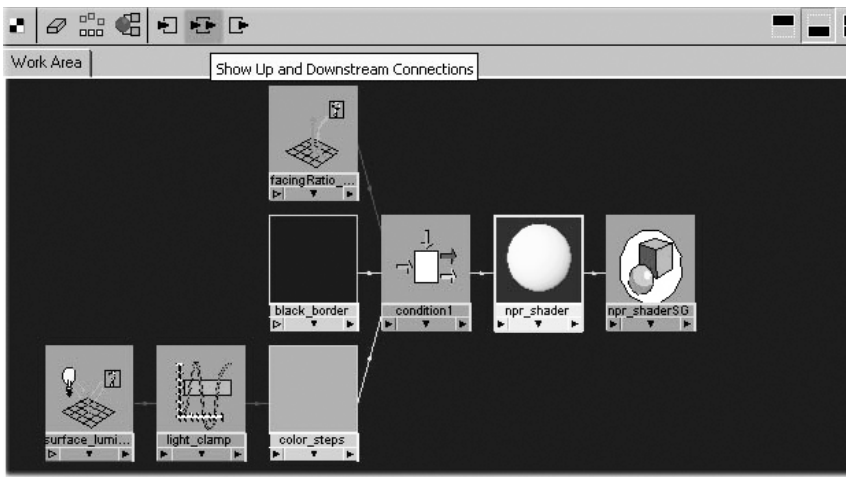


Рис. 11. Шейдер для нефотореалистичного («мультипликационного») освещения

Использование шейдера довольно просто. В пакете Alias Maya, например, он выглядит как сеть из нескольких нод (как было сказано ранее, это обычное представление шейдеров в трехмерных пакетах).

Как правило, такие нефотореалистичные шейдера содержат несколько компонент:

- нода для цвета контура (хотя тут контур некорректен, но используется для правильного совмещения с контуром, полученным другими методами);
- нода для количества так называемых цветовых шагов. В приведенной выше технологии имелось 2 шага — для освещенной и затененной части поверхности. В принципе их может быть больше, вплоть до градиентной заливки;
- нода для вычисления текстурных координат или условий (на рис. 11 она называется condition1).

Могут также присутствовать дополнительные ноды для выбора текстуры в зависимости от освещения, настройки источника освещения, параметров нефотореалистичной тени и т. д.

Работа с таким шейдером довольно проста: мы сначала настраиваем шейдер, те ставим количество шагов изменения цвета (например, для реализации шейдера, реализующего описанный ранее метод закраски трехмерных поверхностей, таких шагов будет 2), настраиваем для них цвета. Расставляем источники освещения на сцену, настраиваем текстуры при необходимости. И в самом конце прикрепляем наш созданный материал к поверхности.

Из недостатков такого и подобных методов закрашивания следует упомянуть то, что при анимации сложных объектов часто приходится увеличивать количество треугольников, составляющих поверхность для придания более сглаженного вида полученному изображению. Это

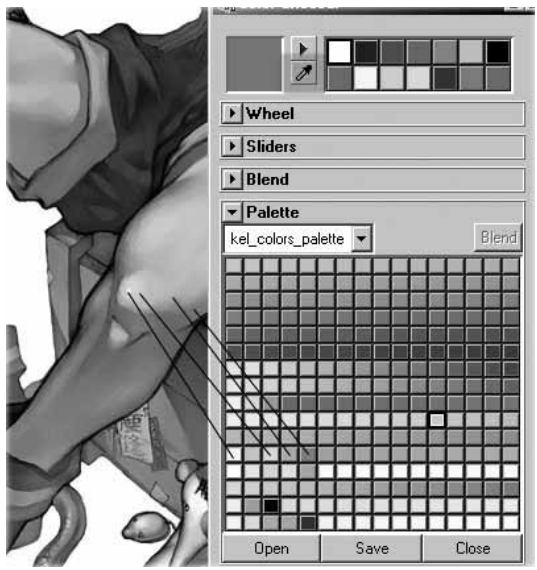


Рис. 12. Окно настройки шейдера для «мультипликационного» освещения

накладывает серьезные ограничения, потому что во многих случаях интерактивная работа со сценой становится невозможна из-за ограничения в скорости сегодняшних видеокарт.

Положительной стороной можно считать, что решений (в том числе бесплатных) этого метода можно найти очень много в Интернете. Метод в большинстве случаев достаточен для легкого создания мультипликационных картин невысокого качества.

В качестве основы для создания как полнометражных, так и фильмов, которые показываются на экране телевизора, приходится или усложнять этот метод для получения более качественного контура, или использовать его вместе с каким-нибудь еще методом. Как правило, в большинстве случаев в России полученные этим способом изображения дорисовываются по кадрам художниками, а также применяются дополнительные методы для получения качественного контура.

3. Другие интересные области в нефотореалистичной графике

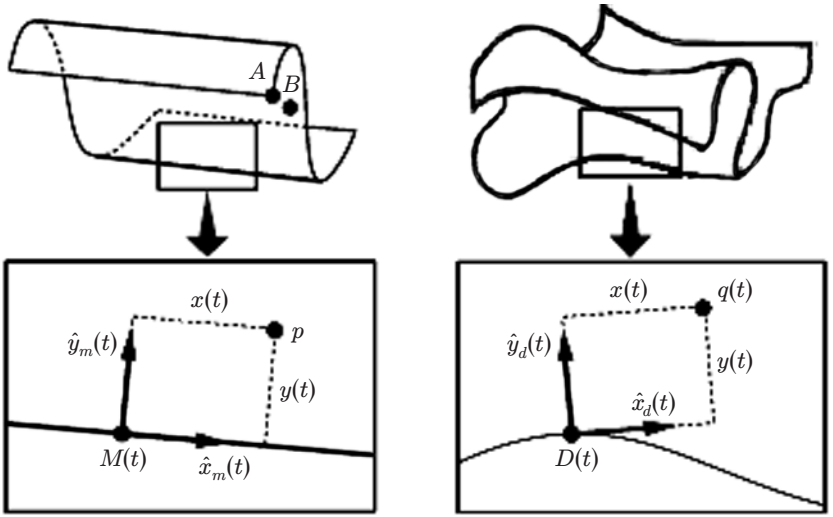
3.1. Деформеры поверхностей

Очень интересной темой является построение поверхностей в мультипликационном виде. Эта технология вкратце описана в [10]. Дело в том, что построение изогнутых поверхностей требует от моделлера довольно большого труда. Как правило, в таких случаях для упрощения работы применяются так называемые «деформеры». Этот механизм позволяет деформировать поверхность и управлять деформацией. В данной работе показана деформация, которая применяется для придания трехмерному объекту вида нарисованного художником.

Входными данными для этого деформера являются: модель, параметры камеры и набор пар (маркер модели — маркер холста) маркерных кривых. Искажение применяется к вертексам модели после того, как она была преобразована в экранную систему координат для данной камеры, но перед тем, как модель будет отрисована. Этот деформер имеет следующие свойства:

1. Каждый маркер модели находится на соответственном ему маркере холста, если смотреть со стороны камеры.
2. Поверхность гладко деформируется между двумя маркерами.
3. Деформер сохраняет кривые в пространстве изображения так, что все деформированные кривые, находящиеся в ракурсе, остаются видимыми.
4. Деформация сохраняет информацию буфера глубины неизменной относительно камеры.

Далее мы кратко опишем процесс искажения одной пары маркеров. Для начала мы возьмем одну пару маркеров в экранных координатах: кривую маркера модели $M(t)$ (которая сгенерирована детектором силуэта)



(a) Model marker curve (b) Drawing marker curve

Рис. 13. Координатная система деформера

и кривую маркера рисования $D(t)$ (полученную от пользователя, который рисует ее вручную).

Для каждой данной точки модели мы можем найти соответственную точку на рисунке, используя соответствующие маркеры. После чего мы можем нарисовать деформированный контур.

Главные особенности этого метода трансформации: он использует гладкие линии, и поэтому искажения будут тоже гладкие, метод также не зависит от таких преобразований как масштабирование, перемещение и поворот, и также важно, что он накладывает маркеры на модель в процессе создания эффекта деформации, а не на постпроцессе.

3.2. Motion Lines

Множество 2-мерных мультфильмов включают негеометрическую информацию в сцене, которая показывает, двигается ли объект или персонаж быстро или медленно, является ли поверхность блестящей и другие эффекты. Линии движения используются в мультфильмах для отображения движения или для отражения стиля художника. На рис. 14 показано, как это выглядит.

Линии рисуются позади движущегося объекта. Без этих дополнительных линий для объектов не было бы зрительного контекста. Этот эффект называется Motion Lines. В нескольких работах для достижения такого эффекта использовался следующий метод: мы используем разницу в положении объекта в двух различных кадрах анимации, затем вычисляем грани, которые оказываются последними по линии движения объекта,

Рис. 14. Линии движения

и рисуем линии, начиная от стыка треугольников и заканчивая некой точкой, которая может, как линейно отставать по обратному вектору движения, так и быть одной из точек сплайна движения объекта.

Эта методика сама по себе двухмерна и дает хорошие результаты только в полностью плоском случае. Как только мы начинает использовать в кадре элементы объема, то линии начинают неправильно передавать перспективу.

В работе [8] использовался усовершенствованный метод. Суть его в следующем. Каждая линия движения — это множество линейных сегментов с множеством ассоциированных с ними параметров (длина, ширина, цвет, стартовая позиция и флаг видимости). Множество этих линий используется для получения видимости движения в сцене. Количество и толщина этих линий может варьироваться в зависимости от стиля рисования. При отрисовке мы знаем, какие объекты в сцене должны двигаться, так что можем дополнительно менять параметры линий в зависимости от параметров движения. Также эти объекты могут задаваться отдельно художником.

В стадии предварительной обработки сцены мы случайным образом выбираем n вертексов трехмерной модели, для которой мы будем рисовать линии движения. Мы выделяем и инициализируем буфер, размером равным как минимум количеству линий движения. В процессе отрисовки мы сохраняем в нашем буфере вектор перемещения, который соответствует перемещению вертекса объекта в текущем кадре, и увеличиваем размер нашего буфера на 1. Для каждой анимационной линии мы проходим через соответственные ему позиции, сохраненные в буфере, и двигаем нашу линию на каждом шаге на вектор перемещения, сохраненный в буфере. В конце мы рисуем сегмент линии между предыдущей и новой позицией нашей линии движения.

В результате нашей обработки мы получаем линии, которые обычно рисует художник и информация о которых не содержится в трехмерной модели. Подобные технологии получения дополнительной информации встречаются достаточно часто.

3.3. Новые решения

Список возможностей достаточно велик. В пакет включены: Недавно компания Cebas Visual Technology, производящая рендер FinalRender (сайт можно найти по адресу: www.finalrender.com) анонсировала новый проект под названием FinalToon. Этот нефотореалистичный рендер предназначен для создания разноплановых изображений в стилизованном виде.

Рис. 15. Результат работы рендера FinalToon

Этот продукт умеет создавать изображения как в рисованном стиле, так и в виде технической иллюстрации и др.

- различные типы линий для технического рисования;
- сглаживание высокого качества;
- 2-мерные эффекты: регулировка силы давления «карандаша», различные типы карандашей и т. д.;
- 3-мерные эффекты: зависимость толщины и прозрачности линий от расстояния до камеры и т. д.;
- закрашивание в «ручном» стиле;
- сохранение настроек каждого материала для наложения на различные объекты;
- линии тени для контура в области тени;
- различные эффекты шума для линий, текстур и т. д.;
- карты граней для использования в качестве контрольных для эффекта «выдавливания», цвета и т. д.;
- отражения и преломления;
- цветовая модель, основанная на температуре объекта.

Стоимость подобной системы оценивается примерно в 500 долл. на одну не сетевую лицензию.

3.4. Технологии нефотореалистичного рендеринга в реальном времени

3.4.1. Реализация при помощи шейдеров

На данный момент развитие графических ускорителей подошло к такому моменту, что на них стали просчитывать сцены с качеством, близком к кинематографическому. Компания NVidia даже назвала свою новую технологию CinematicFX, что означает, что качество картинки, которая

получается этим ускорителями с этой технологией, достигает качества фильма. Неудивительно, что в Интернете появилось огромное число демонстрационных программ, которые показывают, как можно делать нефотореалистичную графику в реальном времени.

Как правило, все эти технологии основаны на специфике оборудования, которое просчитывает картинку. Однако даже это ускорение является достаточным для получения, например, достаточно качественной «мультипликационной» анимации. Дело в том, что использование таких технологий в системах реального времени не накладывает настолько серьезных ограничений на качество как фильм. Действительно, кто будет рассматривать контур, играя в динамичную игру?

Специфика же оборудования на данный момент такова: почти все современные графические ускорители используют технологию шейдинга поверхностей для отрисовки. До февраля этого года можно было использовать только примитивный ассемблер для программирования видеокарт, но с появлением процессоров следующего поколения стало возможным использовать языки и более высокого уровня, как например HLSL (Microsoft), CG (NVidia), а также язык стандарта OpenGL.

В этом разделе мы рассмотрим типичный пример получения нефотореалистичного изображения в реальном времени.

На рис 16 показано окно программы, которая использует модифицированную технологию, раскрытую в [5]. Скорость прорисовки одного кадра на видеокарте NVidia GeForce4 MX составляет 6 кадров. Ниже приведен листинг небольшой программы — вертексного шейдера, который в ней использовался:

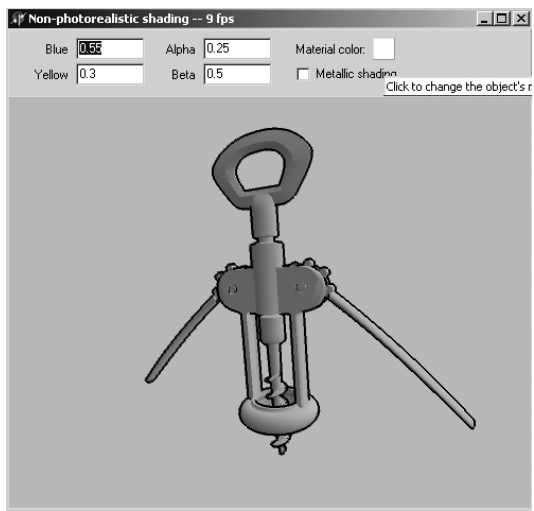


Рис. 16. Нефотореалистичное изображение, полученное в реальном времени

```
!!ARBvp1.0
# Various parameters.
PARAM mvp[4] = { state.matrix.mvp };
PARAM mvit[4] = { state.matrix.modelview.invtrans };
PARAM p[4] = { state.matrix.projection };
PARAM L = { 0.707, 0.707, 0, 1 };
# Hardcoded light for now.
PARAM bias = { 0.5, 0.5, 0.5, 0.5 };
# Shading model parameters.
ATTRIB kd = vertex.color;
PARAM byab = program.local[0];
PARAM blue = { 0, 0, 1, 1 };
PARAM yellow = { 1, 0.8, 0, 1 };
ATTRIB opos = vertex.position;
ATTRIB normal = vertex.normal;
TEMP N, H, lerp, kcool, kwarm, I, kblue, kyellow;
# Transform the normal vector.
  DP4 N.x, mvit[0], normal;
  DP4 N.y, mvit[1], normal;
  DP4 N.z, mvit[2], normal;
  DP4 N.w, mvit[3], normal;
# Calculate cool-to-warm interpolation factor.
  DP3 lerp.x, L, N;
  MAD lerp, lerp.xxxx, bias, bias;
# Calculate cool and warm colors.
  MUL kblue, blue, byab.x;
  MUL kyellow, yellow, byab.y;
  MAD kcool, kd, byab.z, kblue;
  MAD kwarm, kd, byab.w, kyellow;
# Do the lighting.
# Output = LERP((1+L.N)/2, kcool, kwarm);
  ADD I, kwarm, -kcool;
  MAD result.color, lerp, I, kcool;
# Quick 'n dirty specular.
  ADD H, N, L;
  MUL H, H, bias;
  DP3 result.texcoord[0].x, N, H;
  MOV result.texcoord[0].y, lerp.x;
# Transform vertex to clip space.
  DP4 result.position.x, mvp[0], opos;
  DP4 result.position.y, mvp[1], opos;
  DP4 result.position.z, mvp[2], opos;
  DP4 result.position.w, mvp[3], opos;
END
```

Программа реализует модель освещения, разработанную Gooch. Эта модель в основном предназначалась для использования в приложениях технической иллюстрации. Для создания этого эффекта, тени и места, освещенные источником света, конвертировались в отдельный слой, так что важные детали не потерялись в процессе закраски. Освещенные места поверхности преобразовывались в теплый, желтоватый цвет, а тени преобразовывались в голубой. Цвет материала поверхности показывался на границе освещенных и затененных зон.

Цветовая модель реализована повертексно, используя расширение OpenGL GL_ARB_vertex_program. В результате цвет вычислялся в каждом вертексе треугольника, а затем интерполировался по нему. Также была реализована опция, которая разрешала эффект металла на поверхности, который добавлял анизотропную спекулярную компоненту в освещение. Это получалось путем вычисления половинного угла H и использования затем векторного произведения $\mathbf{L}\times\mathbf{N}$ и $\mathbf{N}\times\mathbf{H}$ (где $\mathbf{L}$ — вектор на источник света и $\mathbf{N}$ — нормаль к поверхности) для вычисления текстурных координат текстуры, которая содержит некоторое число диагональных полос.

Параметры цветовой модели могут быть изменены, используя элементы управления в верхней части окна программы. Для более подробного изучения параметров вы можете обратиться на сайт www.delphi3d.net.

В заключение можно сказать, что трехмерная модель обводится черным цветом. Это получается путем отрисовки каркаса модели и использования функции OpenGL glPolygonOffset(), которая меняет значение Z-буфера в процессе отрисовки сцены.

3.4.2. Реализация без шейдеров

Метод, описанный в [12], также занимается отрисовкой нефотореалистичной графики в реальном времени. Дальше будет рассмотрен основной метод, использованный в этой работе.

Этот метод описывает рендеринг силуэтных граней для полигональных поверхностей в пространстве изображения. Важная деталь этого метода, это то, что только 2 набора полигонов необходимо для вычисления видимых силуэтных граней для данной камеры. Эти два набора: P_1 — слой видимых полигонов, находящихся вблизи камеры, и P_2 — второй слой полигонов, удаленных от нее же. Передние грани полигона так же, как и задние грани, хорошо определяют эти два множества. Пересечение P_1 и P_2 дает силуэтные грани в объектном пространстве. Первый слой видимых полигонов может быть просчитан, используя алгоритмы удаления невидимых поверхностей. Для коллекции полигональных моделей замкнутых объектов этот слой делается из передних полигонов, которые полностью видимы (так называемые Front-facing polygons T_1 на рис. 17). Второй слой может быть вычислен, используя такие же алгоритмы удаления невидимых поверхностей, как и в случае первого слоя после вычитания полигонов в первом слое из всех полигонов, находящихся в сцене. Этот слой для замкнутых объектов создается из задних граней

Figure 1. The first layer, *P*1, for scene (a) is made up of parts of nearest facing polygons (b). The second layer, *P*2 is made up of parts of back facing polygons (c). The intersection of these two layers in image space creates silhouette edges (d).

Рис. 17. Иллюстрация к методу

поверхности (см. рис. 17, 1 с). Если мы поместим камеру вне замкнутого объекта, то пересечение P_1 и P_2 даст нам необходимый силуэт.

Визуализировать силуэтные грани возможно путем вычисления проекции P_1, P_2 и $P_1 \& P_2$, используя Z-буфер в пространстве изображения. Места, где значения глубины P_1 и P_2 одинаковы, соответствуют проекции множества $P_1 \& P_2$. Следуя этому можно написать псевдокод для отрисовки силуэтных граней черным цветом на белом фоне. В псевдокоде участвует функция глубины F, которая действует следующим образом: пиксель с глубиной d_1 перезаписывает текущий пиксель в буфере кадра с глубиной d_2, если $F(d_1, d_2) = \text{true}$.

1. Рисуем фон белым цветом.
2. Разрешаем отсечение обратных граней, устанавливаем функцию глубины в (меньше чем).
3. Отрисовываем полигоны (передние) белым.
4. Разрешаем отсечение передних граней, устанавливаем функцию глубины в (равно).
5. Отрисовываем полигоны (задние) черным.
6. Повторяем для новой позиции камеры.

Этот метод, однако, имеет недостатки. В первую очередь недостаточное разрешение обычного Z-буфера может привести к тому, что контур будет теряться. Однако силуэт при этом будет как минимум в один пиксель шириной.

4. Использование в комплексных системах

В данной статье мы не будем рассматривать использование алгоритмов в системах реального времени. Причина проста — эти системы не рассчитаны на использование достаточно сложных схем построения изображений и, как правило, такие алгоритмы специально проектируются под конкретную архитектуру.

Как можно было видеть выше, алгоритмов, которые применяются при построении нефотореалистичных изображений очень много. Для применения и совмещения результатов работы разных алгоритмов необходимо их максимально отделить друг от друга. Для этого создаются отдельные модули, которые могут общаться между собой по унифици-

рованному интерфейсу, а также работать отдельно, наподобие фильтров изображений.

Эти модули должны управляться менеджером рендера, который указывает порядок выполнения вычислений (через граф), а также предоставляет модулям необходимые данные. При этом сам граф выполнения должен быть настраиваемым, чтобы получить максимальную гибкость.

Разделим алгоритмы на классы:

1. Алгоритмы, которые используются художники на этапе моделирования трехмерных сцен (примером могут служить деформеры поверхностей). Этот класс алгоритмов не относится непосредственно к рендерингу.
2. Алгоритмы построения контура. Сюда входят как растровые, так и векторные алгоритмы.
3. Закрашивание (тонирование). Этот класс алгоритмов для растеризации изображений по трехмерным моделям.

Деление 2 и 3 вызвано тем фактом, что, как правило, в существующих системах контур находится отдельно и накладывается на тонированное изображение на этапе постпроцессинга.

В определенных случаях (например, при использовании алгоритмов получения векторного контура) требуется просчитывать контур по Z-буферу. Для решения этой проблемы в интерфейс, по которому взаимодействуют модули нужно внести получение как информации о сцене, так и полученных после растеризации изображениях. При этом ветки выполнения всего процесса рендеринга в таких местах можно распараллеливать. В приведенном случае можно одновременно считать контур и проводить растеризацию. Следующим шагом будет обработка контура по полученным растровым данным.

Наиболее сложными являются случаи совмещения 3- и 2-мерной информации. В этом случае необходимо предоставить выбор художникам для получения нужного результата. Частично проблему можно решить, используя настройки модулей, которые стоит вынести в интерфейс. При этом каждый модуль должен иметь возможность получать информацию о глобальных настройках сцены, таких как параметры камеры, разрешение картинки на выходе и т. д. Такая унификация позволит использовать большинство приведенных выше алгоритмов вместе.

Можно привести пример работы. Предположим, что у нас есть растеризатор, который занимается отрисовкой (возможно с текстурированием, освещением и т. д.) сцен. Также у нас есть векторные и растровые модули для получения контура. С помощью данной системы мы сможем получить изображение с разным контуром, подключая разные модули к выходу растеризатора.

Можно сделать вывод о том, что такая система сможет объединить различные нефотореалистичные алгоритмы в один интегрированный пакет. Предполагается, что такой пакет будет прорывом в данной области и позволит улучшить процесс производства графики.

5. Результаты

На данный момент разрабатывается система, основанная на принципах, описанных выше. Полученные уже сейчас результаты говорят о том, что система легко модернизируется, обладает возможностью параллельной обработки и легко настраивается.

5.1. Глоссарий терминов

Анти-антиалиасинг. Способ обработки (интерполяции) пикселов для получения более четких краев (границ) изображения (объекта). Наиболее часто используемая техника для создания плавного перехода от цвета линии или края к цвету фона. В некоторых случаях результатом является смазывание (blurring) краев.

Краевой антиалиасинг. Краевой антиалиасинг — механизм борьбы с лестничным эффектом. Краевой антиалиасинг сглаживает края полигонов.

Прямые и обратные грани (front and back faces). Front Faces — грани, скалярное произведение нормали к грани которых с вектором направления камеры имеет значение больше 0, т. е. они находятся «видимой» стороной к нам. Обратные грани соответственно «невидимы».

Спекулярная характеристика света. Световая характеристика, которая определяет то, как свет будет отражаться от объектов.

Вертекс. Точка в трехмерном пространстве. То же самое, что и вектор.

Шейдер. Небольшая программа на специализированном языке, которая в общем случае описывает свойство материала поверхности.

***Z*-буфер.** Часть графической памяти, в которой хранятся расстояния от точки наблюдения до каждого пиксела (значения Z). Z-буфер определяет, какая из многих перекрывающихся точек наиболее близка к плоскости наблюдения. Также, как большее число битов на пиксель для цвета в буфере кадра соответствует большему количеству цветов, доступных в системе изображения, так и количество бит на пиксель в Z-буфере соответствует большему числу элементов. Обычно, Z-буфер имеет не менее 16 бит на пиксел для представления глубины цвета. Аппаратные акселераторы 3D-графики могут иметь собственный Z-буфер на графической карте, чтобы избежать удвоенной нагрузки на системную шину при передаче данных. Некоторые реализации Z-буфер используют для хранения не целочисленное значение глубины а значение с плавающей запятой от 0 до 1.

Литература

1. Interactive Non-Photorealistic technical illustration. Amy Gooch. The University of Utah, 1998. December.
2. Aaron Hertzmann. Introduction to 3D Non-Photorealistic Rendering: Silhouettes and Outlines. SIGGRAPH 99 Course on Non-Photorealistic Rendering. Los Angeles, 1999. Aug.

3. *Bruce Gooch*, *Peter-Pike J. Sloan*, *Amy A. Gooch*, *Peter Shirley*, *Richard Riesenfeld*. Interactive Technical Illustration // Stephen N. Spencer (ed.). Proceedings of the Conference on the 1999 Symposium on Interactive 3D Graphics. N. Y., ACM Press. 1999. Apr. P. 31–38.
4. *George Winkenbach*, *David H. Salesin*. Rendering Parametric Surfaces in Pen and Ink // Holly Rushmeier (ed.). Proceedings of SIGGRAPH'96 (New Orleans, August 1996), Computer Graphics Proceedings, Annual Conference Series. N. Y., 1996. ACM SIGGRAPH 1996. Aug. P. 469–476.
5. *Amy Gooch*, *Bruce Gooch*, *Peter Shirley*, *Elaine Cohen*. A Non-Photorealistic Lighting Model for Automatic Technical Illustration // Michael Cohen (ed.). Proceedings of SIGGRAPH'98 (Orlando, July 1998), Computer Graphics Proceedings, Annual Conference Series. N. Y., 1998. ACM SIGGRA. P. 447–452.
6. *Amy Ashurst Gooch and Peter Willemsen*. Evaluating Space Perception in NPR Immersive Environments. Proceedings of the 2^{nd} international symposium on Non-Photorealistic animation and rendering (NPAR 2002). Annecy, France. 2002. P. 105–110.
7. *Lee Markosian*, *Michael A. Kowalski*, *Samuel J. Trychin*, *Lubomir D. Bourdev*, *Daniel Goldstein*, *John F. Hughes*. Real-Time Non-Photorealistic Rendering // Turner Whitted (ed.). Proceedings of SIGGRAPH'97 (Los Angeles, August 1997), Computer Graphics Proceedings, Annual Conference Series. P. 415–420. ACM SIGGRAPH, 1997.
8. *Adam Lake*, *Carl Marshall*, *Mark Harris*, *Marc Blackstein*. Stylized Rendering Techniques for Scalable Real-Time 3D Animation / Proceedings of NPAR 2000, Symposium on Non-Photorealistic Animation and Rendering (Annecy, France, June 2000). P. 13–20. N. Y., 2000. ACM.
9. *Allison W. Klein Wilmot Li*, *Michael M. Kazhdan*, *Wagner T. Correa*, *Adam Finkelstein*, *Thomas A. Funkhouser*. Non-Photorealistic Virtual Environments / Kurt Akeley (ed.). Proceedings of SIGGRAPH 2000 (New Orleans, July 2000). Computer Graphics Proceedings. Annual Conference Series. P. 527–534. N. Y., 2000. ACM SIGGRAPH. Wagner Toledo Correa, Robert J. Jensen, Craig E. Thayer, Adam Finkelstein.Texture Mapping for Cel Animation. Princeton University.Walt Disney Feature Animation. SIGGRAPH 98 Conference Proceedings, Annual Conference Series, pages 435-446. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.
10. *Wagner Toledo Correa*, *Robert J. Jensen*, *Craig E. Thayer*, *Adam Finkelstein*. Texture Mapping for Cel Animation / Princeton University. Walt Disney Feature Animation.
11. Edge-Enhancement — An Algorithm for Real-Time Non-Photorealistic Rendering / Marc Nienhaus, Juergen Doellner. Hasso-Plattner-Institute for Software, Engineering at the University of Potsdam.
12. Image Precision Silhouette Edges / Ramesh Raskar, Michael Cohen. University of North Carolina at Chapel Hill, Microsoft Research.
13. WYSIWYG NPR: Drawing Strokes Directly on 3D Models / Robert D. Kalnins, Lee Markosian, Barbara J. Meier, Michael A. Kowalski, Joseph C. Lee, Philip L. Davidson, Matthew Webb, John F. Hughes, Adam Finkelstein. Princeton University, Brown University.
14. Hardware Support for Non-Photorealistic Rendering. Ramesh Raskar. Mitsubishi Electric Research Labs, Cambridge, MA, Siggraph Graphics Hardware, 2001, Aug. Los Angeles, CA, USA, 2001.

15. Non-Realistic Rendering. API. Steve Kilthau Graphics Hardware, 2001, Aug. Los Angeles, CA, USA, 2001.
16. OpenGL Reference Manual. OPENGL Architecture Review Board. Addison-Wesley: Developers Press, 1996.
17. Extensions specification documents. OPENGL, ARB. 1999. Mar. (opengl.org/Documentation/Extensions.html).
18. DirectX SDK (ver 3.0, 5.0) Reference. Microsoft Corporation, 1997.