Виртуальная реальность

занятие №5

Рябинин Константин Валентинович

e-mail: icosaeder@ya.ru
jabber: icosaeder@jabber.ru

Пермь, 2012

Анимация

Анимация – это вдыхание жизни в сцену :)

Анимация – это последовательное отображение кадров с различным содержанием

(конструктивно)

Программно-аппаратная поддержка анимации:
механизм двойной буферизации

→ **Строго говоря, любое свойство объекта может быть анимировано**

Важное требование к анимации:
сохранение межкадрового соответствия
(*frame-to-frame coherence*)

Скорость анимации

Анимация по таймеру:

- Идея: запуск некоторого таймера (должен поддерживаться системой), который в наперёд заданные моменты времени вызывает функцию обновления состояния
- Способ имеет право на существование лишь в исключительных случаях
- Таймер лучше использовать как триггер анимации, а не как её «драйвер»

Непрерывная анимация:

- Идея: обновление состояния происходит с максимально возможной скоростью, а величина изменения вычисляется на основе желаемой и фактической скорости

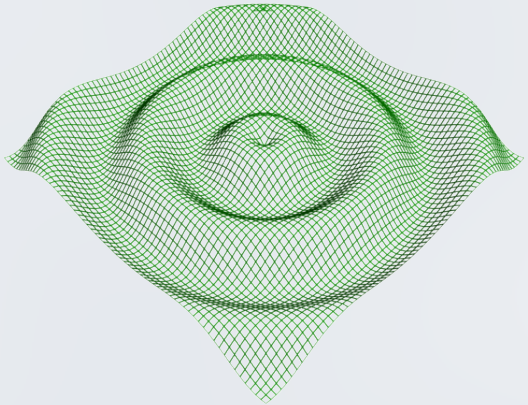
Поверхности

- Чаще всего **трёхмерные объекты** представляются лишь своими **внешними поверхностями**

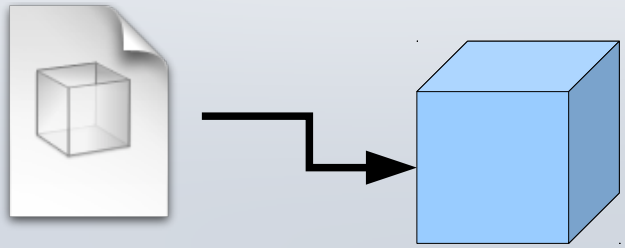
Подходы к заданию поверхностей

- Аналитически (процедурные поверхности)

$$z(x;y)=a\cdot\sin\left(\alpha\left(x^2+y^2\right)\right)\cdot e^{-\beta\left(x^2+y^2\right)}$$



- Наперёд заданным массивом вершин (загрузка из файлов)



Трёхмерные объекты

Семантический разрыв между проектирование / обработкой сцены и её отображением:

- **Прикладной программист стремится оперировать трёхмерными объектами, тогда как визуализация производится на уровне множества полигонов.**

  То есть на низком уровне всякое объектное представление сцены теряется

→ **Часто необходимы соответствующие прослойки для хранения структур объектов; естественным является использование ООП**

Квадрик-примитивы

Квадрика – это алгебраическое многообразие, которое можно задать однородным квадратным уравнением

Квадрика – это поверхность 2-го порядка

Квадрик-примитив – это трёхмерный объект на основе полигонализации поверхностей второго порядка

Характерные черты:

- **Чаще всего являются процедурными**
- **Имеют легко управляемую детализацию**
- **Основное назначение: конструирование тестовых сцен**

Освещение

Источник света – это объект или псевдообъект в трёхмерном пространстве, характеризующий положение и свойства осветителя

Модель освещения – конкретная схема определения интенсивности отражённого к наблюдателю света в каждой точке изображения

Виды моделей освещения:

- **Локальные – во внимание принимается только свет, падающий на поверхность объекта от источников**
- **Глобальные – во внимание принимается свет, падающий на поверхность объекта от источников и отражённый от других объектов сцены**

Освещение

Критерий качества модели освещения – не физическая точность, а **визуальный эффект**

Любая модель освещения должна учитывать кривизну поверхности. Учёт происходит при помощи **нормалей**

- Чаще всего (в частности, в OpenGL) нормали задаются **в вершинах** поверхности

Материал – совокупность визуальных свойств поверхности

Модель освещения характеризуется:

- Допустимыми свойствами источника света
- Допустимыми свойствами материала объекта
- Алгоритмом вычисления интенсивности отражённого к наблюдателю света

Освещение в OpenGL

Мы будем рассматривать только локальные модели

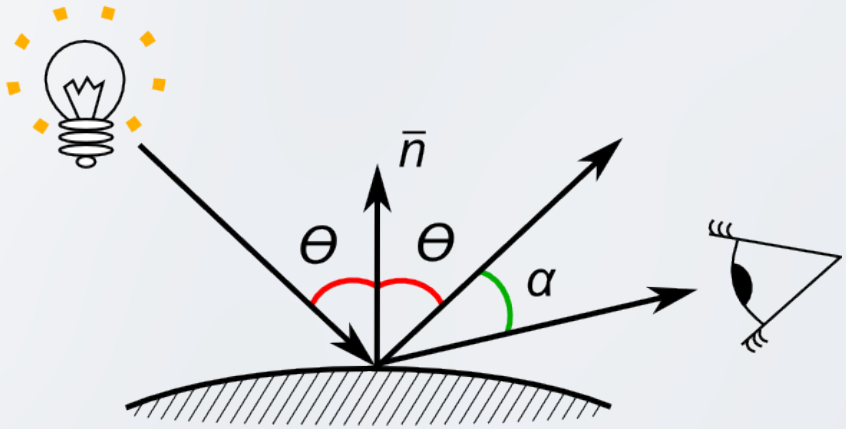
Важные понятия модели освещения в OpenGL:

(являются важными в большинстве существующих простых моделей)

- **Диффузное освещение (diffuse) – свет от источника, равномерно рассеянный поверхностью во всех направлениях (матовая поверхность)**
- **Зеркальный блик (specular) – свет от источника, отражённый от поверхности в конкретном направлении (зеркальная поверхность; конкретным является направление по оси зрения наблюдателя)**
- **Окружающее освещение (ambient) – свет «отражённый от окружающей обстановки», равномерно рассеянный поверхностью во всех направлениях**

- **Ослабевание света (attenuation) – уменьшение интенсивности света по мере удаления от источника**

Освещение в OpenGL



$$I = I_a \cdot K_a + \frac{I_d \cdot K_d \cdot \cos\theta + I_s \cdot K_s \cdot \cos^N \alpha}{C_s + C_l \cdot D + C_n \cdot D^2}$$

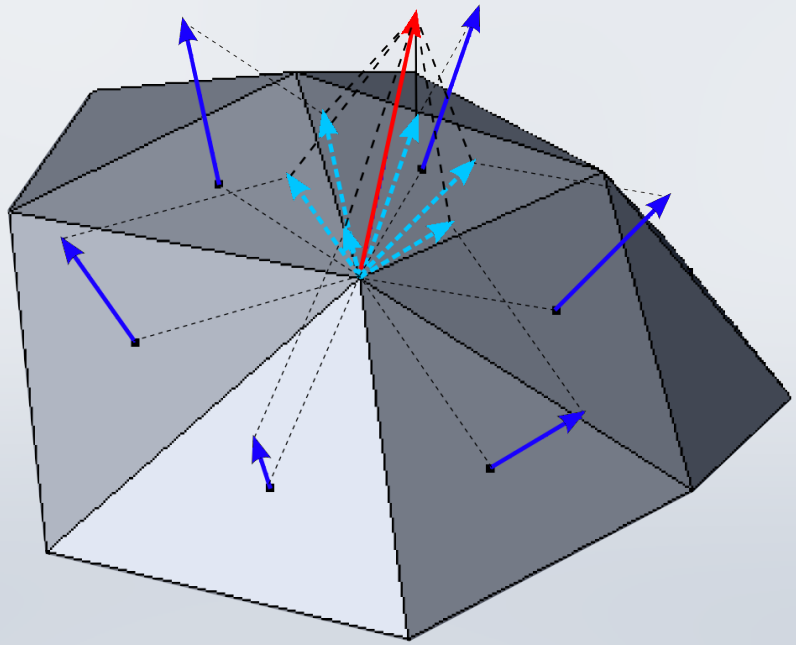
I – итоговый цвет
I_a – цвет окружающей подсветки источника света
K_a – цвет окружающей подсветки материала
I_d – цвет диффузного освещения источника света
K_d – цвет диффузного освещения материала
I_s – цвет зеркального блика источника света
K_s – цвет зеркального блика материала
N – показатель величины блика
C_c – константне затухание
C_l – коэффициент линейное затухание
C_n – коэффициент квадратичного затухания
D – расстояние от источника света до точки поверхности

Способы закраски полигонов

- **Плоская закраска – полигон закрашивается цветом, являющимся средним арифметическим цветов в его вершинах**
- **Закраска Гуро – цвет в каждой точке полигона есть результат билинейной интерполяции цвета вдоль сканирующей строки**
- **Закраска Фонга – цвет в каждой точке полигона перевычисляется на основе нормали, полученной билинейной интерполяцией вдоль сканирующей строки**

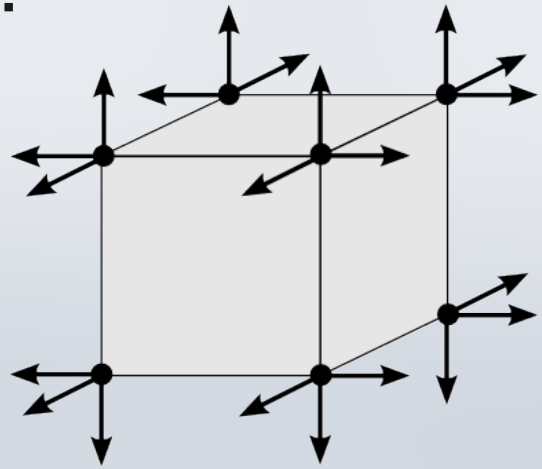
Вычисление нормали в вершине

- Если поверхность объекта задаётся аналитически, вектор нормали в каждой вершине является градиентом порождающей функции
- Если поверхность объекта задаётся множеством наперёд заданных многоугольников, у которых известны только координаты вершин, наиболее правильным способом вычисления нормали в каждой вершине является среднее арифметическое нормалей к многоугольникам, смежным с данной вершиной (алгоритм сглаживающих групп)



Вычисление нормали в вершине

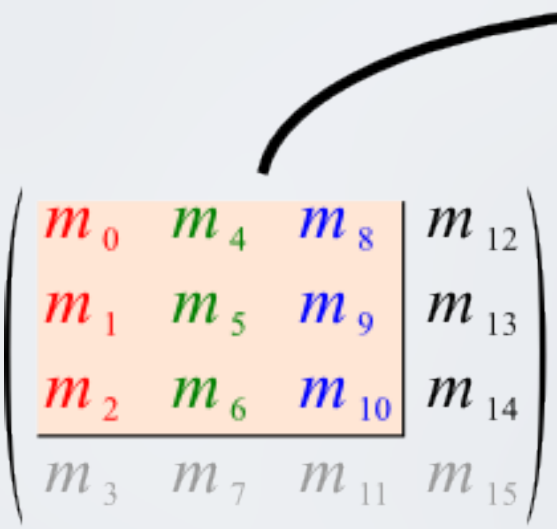
- В результате вычисления нормалей по алгоритму сглаживающих групп, светотень по поверхности распространяется плавно (что вызывает эффект гладкой поверхности)
- Данный эффект не всегда является желательным: иногда следует вывести «гранёную» поверхность
- Для этого может быть использована «плоская» закраска (flat shading), но её использование не всегда возможно (в новых версиях OpenGL она не поддерживается)
- В этом случае необходимо дублировать данные о нормалях в вершинах:



Трансформация нормалей

- Нормаль принято задавать направлением (вектором)
- Перед расчётом освещённости, нормаль необходимо нормировать
- В процессе применения преобразований вершин, перпендикулярность этого направления плоскости грани, которой инцидентна вершина с данной нормалью, может быть нарушена
- При проецировании на плоскость экрана перпендикулярность нарушится в любом случае
- Для решения этих проблем:
  - Расчёт освещённости делают для вершины, только лишь размещённой на сцене, но ещё не спроектированной на экран (то есть трансформированной только матрицей ModelView)
  - Трансформацию нормали производят не матрицей ModelView, а особой матрицей, которая является инверсно-транспонированным первым главным минором 3x3 матрицы ModelView

Трансформация нормалей



$$N = \left(\begin{pmatrix} m_0 & m_4 & m_8 \\ m_1 & m_5 & m_9 \\ m_2 & m_6 & m_{10} \end{pmatrix}^{-1} \right)^{T}$$

$$n' = N \cdot n$$