* OpenGL – это спецификация, определяющая независимый от языка программирования платформонезависимый программный интерфейс для написания приложений, использующих двумерную и трёхмерную компьютерную графику.

(Википедия)

* Конвейер, который включает в себя несколько программируемых этапов, и несколько фиксированных.

(Консорциум)

* Specify geometry using primitives: triangles, quadrilaterals, lines, points, etc…
* К контексту OpenGL принадлежит два цветных буфера, один для отрисовки, другой для отображения его на экране. SwapBuffers() меняет их местами
* На самом деле буферов, в том числе и цветных больше, ещё есть буфер глубины, маски.
* Перед отрисовкой каждого кадра эти буферы надо очистить с помощью glClear();

**Видовая трансформация**

* + - gluLookAt (eyeX, eyeY, eyeZ, centerX, centerY, centerZ, upX, upY, upZ)
* По умолчанию
  + Позиция камеры (0, 0, 0)
  + Направление вдоль отрицательного направления оси z
  + Вектор верхнего направления (0, 1, 0)

**Модельная трансформация**

* + Translation: glTranslate(x,y,z)
  + Scale: glScale(sx,sy,sz)
  + Rotation: glRotate(theta, x,y,z)

**Проекционная трансформация**

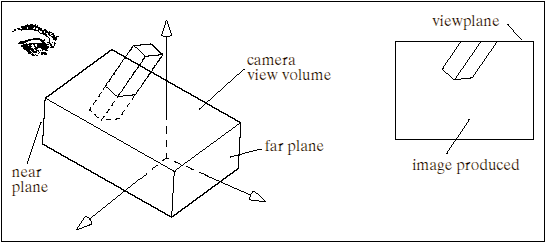
* Перспективная проекция
* Ортографическая проекция

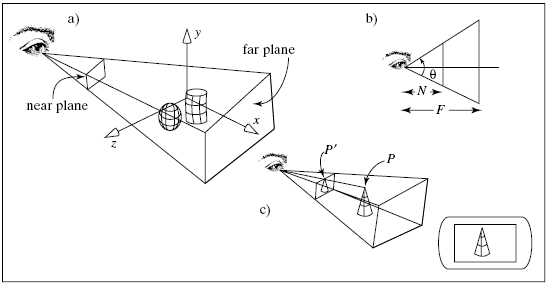
**Трансформация порта просмотра**

* Задает форму и размеры доступной области на экране, куда будет перенесено изображение
* glViewport() задает начальную точку доступного экранного пространства внутри окна, а также ширину и высоту доступной области на экране.

Модельно-видовая матрица

* + Видовые и модельные преобразования в OpenGL объединены в одной матрице.
  + для достижения определенной композиции вы можете либо перемещать камеру, либо перемещать все объекты сцены в противоположном направлении.
  + Модельное преобразование, поворачивающее объекты сцены против часовой стрелки аналогично видовому преобразованию, которое поворачивает камеру по часовой стрелке.
  + Команды видового преобразования должны вызываться перед всеми командами модельных преобразований, чтобы модельные преобразования были применены к объектам первыми.

Orthographic projection

Perspective projection

* Intrinsic (optical) properties of camera:

center

near

fov

z

x

y

Освещение

* Метод трассировки лучей, и метод излучательности требуют большого объема вычислений, поэтому основное внимание уделяется более простым локальным моделям заполнения, основанным на модели отражения Фонга (Phong)

Типы взаимодействия света и материала поверхности

* Зеркальное отражение. Поверхности выглядят блестящими, т.к. большая часть световой энергии отражается или рассеивается в узком диапазоне углов, близких к углу отражения.
* Диффузное отражение. При диффузном отражении падающий свет рассеивается в разных направлениях.
* Преломление. Луч света, падающий на поверхность, преломляется и проникает в среду объекта под другим углом. Как правило, при этом отражается часть падающего света.

Источник

* любой источник рассматривается, как состоящий из трех независимых источников первичных цветов и соответственно его описывает трехкомпонентная функция излучения:



Освещение

* OpenGL рассчитывает свет и освещение так, как будто свет может быть разделен на красный, зеленый и синий компоненты.
* источник света характеризуется количеством красного, зеленого и синего света, которое он излучает
* материал поверхности характеризуется долями красного, зеленого и синего компонентов, которые он отражает в различных направлениях.

4 компоненты освещения (задается материалом)

* + *фоновое* (*ambient)* свет, который настолько распределен в среде, что его направление определить невозможно. Когда фоновый свет падает на поверхность, он одинаково распределяется во всех направлениях
  + *диффузное (diffuse)* свет, идущий из одного направления, таким образом, он выглядит ярче, если падает на поверхность под прямым углом, и выглядит тусклым, если касается ее всего лишь вскользь. Когда он падает на поверхность, он распределяется одинаково во всех направлениях
  + *зеркальное (specular)* исходит из определенного направления и отражается от поверхности в определенном направлении.
  + *исходящее* (эмиссионное – *emissive)* свет, исходящий от самого объекта. Добавляет объекту интенсивности, но на него не влияют никакие источники света, и он не производит дополнительного света для сцены в целом*.*

Материал

* материалы имеют разные фоновый, диффузный и зеркальный цвета, которые задают реакцию материала на фоновый, диффузный и зеркальный компоненты света.
* Фоновый цвет материала комбинируется с фоновым компонентом всех источников света, диффузный цвет с диффузным компонентом, а зеркальный с зеркальным.
* Фоновый и диффузный цвета задают видимый цвет материала, они обычно близки, если не эквивалентны.
* Зеркальный цвет обычно белый или серый.

Источники направленного света

* Источник света такого типа находится в бесконечности и свет от него распространяется в заданном направлении.
* GL\_POSITION по умолчанию (0.0, 0.0, 1.0, 0.0) - направление источника направленного света
* Первые три компоненты *(x, y, z)* задают вектор направления, а компонента *w* всегда равна нулю (иначе источник превратится в точечный).

Точечные источники света

* Точечный источник света расположен в некоторой точке пространства и излучает во всех направлениях.
* закон убывания интенсивности излучения с расстоянием в виде обратно-квадратичной функции от расстояния: fatt(d) = 1/(kconst + klinear\*d+kquadratic\*d\*d)
* Коэффициенты задаются параметрами
  + GL\_CONSTANT\_ATTENUATION
  + GL\_LINEAR\_ATTENUATION
  + GL\_QUADRATIC\_ATTENUATION

Прожекторы

* прожектор позволяет ограничить распространение света конусом
* можно задать коэффициент убывания интенсивности, в зависимости от угла между осью конуса и лучом распространения света

Свойства источника света OpenTK

* GL\_LIGHT\_MODEL\_LOCAL\_VIEWER в графической системе устанавливает режим "близкого" наблюдателя. Если наблюдатель расположен далеко от рассматриваемой сцены, то можно считать, что вектор, задающий направление на наблюдателя, для всех объектов сцены один и тот же. Это сокращает объем вычислений.
* Если наблюдатель с определенной позиции может "заглянуть" внутрь объекта и увидеть внутренние грани, тогда для корректного закрашивания нужно установить в true параметр GL\_LIGHT\_MODEL\_TWO\_SIDED

Свойства источника света

* геометрические параметры источников света преобразуются матрицей вида, поэтому можно задавать их положение, используя привычные средства преобразования.

Глобальное фоновое освещение

* В сцену можно включить и глобальное фоновое освещение, которое не связано ни с каким отдельным источником. Если, например требуется слабо подсветить все объекты сцены белым цветом, в программу следует включить такой фрагмент кода:

GLfloat global\_ambient[]={0.1, 0.1, 0.1, 1.0};

glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, global\_ambient);

Установка параметров материала

* void glMaterial{if}(GLenum face, GLenum pname, TYPE param);
* Аргумент *face* указывает для каких граней объекта задается свойство материала: *GL\_FRONT, GL\_BACK GL\_FRONT\_AND\_BACK*
* GL\_AMBIENT (0.2, 0.2, 0.2, 1.0) цвет фонового отражения материала
* GL\_DIFFUSE (0.8, 0.8, 0.8, 1.0) цвет рассеянного отражения материала
* GL\_SPECULAR (0.0, 0.0, 0.0, 1.0) цвет зеркального отражения материала
* GL\_EMISSION (0.0, 0.0, 0.0, 1.0) цвет собственного излучения материала
* GL\_SHININESS 0.0 степень в формуле зеркального отражения (коэффициент блеска). Допускаются значения в интервале [0; 128].
* GL\_AMBIENT\_AND\_DIFFUSE   цвет фонового и рассеянного отражения материала
* По умолчанию, при включении расчета освещения, текущий цвет, задаваемый командой glColor4f, игнорируется.
* Можно включить управление свойством материала с помощью текущего цвета, т.е. изменять одну из характеристик отражения материала командой glColor4f()
* glEnable(GL\_COLOR\_MATERIAL);
* glColorMaterial(GLenum face, GLenum mode); задает, какое конкретно свойство материала будет передаваться текущим цветом

OpenGL: Flat Shading

* В режиме плоского закрашивания OpenGL использует вектор нормали, ассоциированный с первой вершиной каждого очередного закрашиваемого многоугольника
* glShadeModel(GL\_FLAT);
* OpenGL будет интерполировать цвет вдоль отображаемого примитива.
* glShadeModel(GL\_SMOOTH);

Текстуры

* Текстуры – это прямоугольные массивы данных, например, цветовых, световых или цветовых и альфа. Индивидуальные элементы (значения) текстуры часто называются *текселями (texels).*
* *Что делает текстурирование сложным, так это то, что* прямоугольная текстура может быть наложена на непрямоугольный объект, и это должно быть сделано каким-либо разумным способом.

Наложение текстуры

* Чтобы использовать наложение текстуры, вы должны выполнить следующие шаги:
  + Создать текстурный объект и задать текстуру для него
  + Задать, как текстура должна воздействовать на каждый пиксель
  + Активизировать механизм текстурирования
  + Нарисовать сцену, передавая на конвейер визуализации и геометрические координаты и координаты текстуры
* Текстурирование работает только в RGBA режиме. Результат попытки применения текстурирования в индексном режиме не определен.

Загрузка текстуры в память

* + void glTexImage(1/2/3)D(GLenum target, GLint level, GLint internalFormat, GLsizei width, (GLsizei height), GLint border, GLenum format, GLenum type, const GLvoid \* data)
  + **target**: GL\_TEXTURE\_(1/2/3)D
  + **level** : загруженный уровень сокращенной текстуры (mipmap), для обычной текстуры равен 0
  + **internalFormat**: сколько компонентов цвета на тексель нужно записывать.
  + **width**, **height** : размеры текстуры, должны быть степенями 2
  + **border** : должен быть равен 0 для одномерной текстуры.
  + **format**: формат данных пикселя. Допустимые значения: GL\_RED, GL\_RG, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_RED\_INTEGER, GL\_RG\_INTEGER, GL\_RGB\_INTEGER, GL\_BGR\_INTEGER, GL\_RGBA\_INTEGER, GL\_BGRA\_INTEGER, GL\_STENCIL\_INDEX, GL\_DEPTH\_COMPONENT, GL\_DEPTH\_STENCIL.
  + **type** : тип данных пикселя. Допустимые значения: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.
  + **data**: указатель на данные изображения в памяти

Текстурные координаты

* Информация о том, как наложить текстуру на геометрический объект
* Для этого задаются *текстурные координаты* каждой вершины в диапазоне от 0 до 1
* void glTexCoord2f(GLfloat s, GLfloat t);
* текстурные координаты надо задать ДО координат вершины

(0,0)

(1,0)

(0.5, 1)

(0,0)

(1,1)

(0,1)

(1,0)

Вычисление итогового цвета

* То, как OpenGL объединяет цвета текселей с цветом геометрического объекта, на который накладывается текстура, зависит от режима текстурной среды.
* void glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GLfloat param);
* param:
  + - GL\_ADD складывает цвет текстуры и цвет текселя
    - GL\_MODULATE цвет текселя умножается на цвет геометрического объекта
    - GL\_DECAL аналогично GL\_REPLACE для текстур без прозрачности
    - GL\_BLEND смешивание
    - GL\_REPLACE цвет текселя заменяет цвет объекта
    - GL\_COMBINE

Фильтрация текстуры

* Между текселями и пикселами почти никогда не бывает взаимно-однозначного соответствия
* процесс расчета растянутой или сжатой картой текстуры называется *фильтрацией*
* void glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GLfloat param);
  + - GL\_NEAREST
    - GL\_LINEAR
    - GL\_NEAREST\_MIPMAP\_NEAREST
    - GL\_LINEAR\_MIPMAP\_NEAREST
    - GL\_NEAREST\_MIPMAP\_LINEAR
    - GL\_LINEAR\_MIPMAP\_LINEAR

Намотка текстуры

* Если текстурные координаты выходят из диапазона [0, 1], то OpenGL обрабатывает их согласно текущему режиму *намотки,* его можно установить отдельно для каждой координаты
* void glTexParameteri(GL\_TEXTURE\_2D GL\_TEXTURE\_WRAP\_S , GLfloat param);
* void glTexParameteri(GL\_TEXTURE\_2D GL\_TEXTURE\_WRAP\_T , GLfloat param);
  + GL\_CLAMP\_TO\_EDGE
  + GL\_MIRRORED\_REPEAT
  + GL\_REPEAT

Текстурные объекты

* Переиспользование загруженной текстуры
  + Сгенерируйте имена текстур.
  + Привяжите объекты текстуры к данным текстуры (в частности к массивам изображений и свойствам).
  + Выбирайте (повторно связывайте) ваши текстурные объекты, делая их текущими для визуализации текстурированных моделей.

Именование текстурных объектов

* В качестве имени текстуры может быть использовано любое ненулевое беззнаковое целое. Для получения неиспользуемых имен текстур используйте **glGenTextures()**.
* void **glGenTextures (GLsizei *n, GLuint \*textureNmes);***
* **glBindTexture()** используется и при создании, и при использовании текстурных объектов.
  + - При начальном связывании создается новый текстурный объект.
    - Когда объект текстуры связывается впоследствии, данные, содержащиеся в нем, становятся текущим состоянием текстуры, замещая предыдущее состояние.

Очистка текстурных объектов

* void **glDeleteTextures (GLsizei *n, const GLuint \*textureNames);***
* Удаляет *n текстурных объектов, чьи имена переданы в аргументе textureNames.* Освобожденные имена могут быть повторно использованы **Если удаляется текущая текстура, состояние переходит к** текстуре по умолчанию с *textureName* равным 0. Попытки удаления имен несуществующих текстур игнорируются без генерации каких-либо ошибок.

Автоматическое генерирование текстурных координат

* void **glTexGen{ifd} (GLenum *coord, GLenum pname, TYPE param);***
* void **glTexGen{ifd}v (GLenum *coord, GLenum pname, TYPE \*param);***
* *coord: GL\_S, GL\_T, GL\_R*
* *pname может* принимать значения GL\_TEXTURE\_GEN\_MODE, GL\_OBJECT\_PLANE, GL\_EYE\_PLANE, GL\_SPHERE\_MAP. Если задано значение GL\_TEXTURE\_GEN\_MODE, *param должен быть* целым числом (или указателем на целое число, если используется векторная версия команды), которое равно GL\_OBJECT\_LINEAR, GL\_EYE\_LINEAR или GL\_SPHERE\_MAP.
* Эти символические константы указывают на то, какая функция должна использоваться для вычисления координат текстуры. Со всеми остальными возможными значениями для *pname, param должен представлять собой указатель на массив величин (в векторной* версии команды), задавая параметры функции вычисления текстурных координат.

Mipmapping

* Сокращенные или множественные текстуры
* Серии предварительно отфильтрованных карт текстуры с разными разрешениями (128x128, 64x64, 32x32 и так далее)