Вводная лекция

5 сентября 2015 г.
10:05

Темы лабораторных работ:

1. Графические примитивы
    a. В C# переопределить метод OnPaint в классе Forms
2. Проектирование графиков (нужно сдать до середины октября)
3. Проектирование в двумерных пространствах
4. Проектирование в трёхмерных пространствах (надо закончить до ноября)
5. OpenGL (можно начинать делать РГЗ)
6. РГЗ по OpenGL

VC++, C++, C#

Лучше писать на VC++, потому что там нет классов.

Лабы до зачётной недели, РГЗ на ней.

Краткая история развития компьютерной графики

Компьютерная графика — это наука, которая изучает методы преобразования объектов в изображения. Формирование компьютерной графики как науки связывают с появлением специального пакета Сазерленда в начале 60-х годов. Он написал программу Sketchpad, позволявшую создавать простые трёхмерные объекты.

В 60-е годы. Появляются алгоритмы формирования теней, удаления невидимых поверхностей. Первые рабочие станции появились в конце 70-х годов и объединили лучшие технологии: работа с графическими устройствами, организация ввода-вывода, организация связи (сетевое взаимодействие).

В 80-е годы. С появлением персональных компьютеров появились принципы построения реальных объектов.

86-90-е годы. Появление технологии мультимедиа. К графике добавились обработка звука и видеоизображение, общение пользователя с компьютером расширилось.

90-10-е годы. Появление реалистичной графики, имитирующей реальный мир. Физика твёрдого тела, механика сплошных сред: гидродинамика, газы, частицы и пр. Освещение, т. е. моделирование законов оптики.

Наши дни. Очки виртуальной реальности, системы дополненной реальности.

Суперстанции

Суперстанции — это соединение в одной системе возможностей работы рабочих станций (3D-графика) и суперкомпьтеров (быстрый ввод-вывод и векторизация вычислений).

Типовую суперстанцию можно описать как систему, состоящую из следующих компонент:

1. Центрального процесса с одним или несколькими ядрами.
2. Сопроцессор (сопроцессор с плавающей запятой; возможно векторный).
3. Графический процессор с кадровым буфером и Z-буфером.
4. Не менее, чем 32-битная система.
5. Сетевой контроллер.

6. Внешняя память и оперативная память.
7. Стандартная шина ввода-вывода для подключения периферийных устройств.
8. Внешние устройства: клавиатура, мышь, монитор.

Рабочие станции развиваются более динамично, чем другие классы компьютеров, т. к. в рабочих станциях применяются современные достижения в области процессоров, графики операционных систем. Указанные процессы приводят к следующему: возрастает общая производительность рабочих станций; увеличивается периферийные возможности; улучшается ПО и эргономические свойства.

Растровая графическая дисплейная система

РГС осуществляет функции формирования и модификации наборов данных в видеопамяти и управление режимами вывода графической информации на монитор. Функциональные компоненты РГС:

1. Видеопамять — служит для хранения графических данных в растровой форме.
2. Графический процессор — реализует основные функции по формированию изображения в видеопамяти. В современных системах графический процессор выполняет два класса операций: преобразование графических примитивов в растровую форму; копирование прямоугольных блоков в видеопамяти.
3. Видеоконтроллер — формирует управляющие сигналы для организации доступа к видеопамяти со стороны графического и центрального процессора. А также обеспечивает генерацию экранного буфера в видеопамяти. В состав видеоконтроллера как правило входит аппаратура, управляющая графическим монитором, градиентом яркости, схемой границ светлости изображения; средства поддержки атрибутов изображения (мерцания, подсветка, наложение и т. д.)

Общие принципы работы графических приложений

19 сентября 2015 г.
10:07

Выполнение графических приложений в реальном времени накладывает серьезные вычислительные требования на компьютеры. Чтобы поддерживать непрерывное движение для человеческого глаза необходимо сгенерировать по крайней мере 20 FPS. Одно изображение состоит приблизительно из миллиона пикселей, что означает для одного пикселя цвет должен быть вычислен меньше, чем за 50 нс. В такие короткие сроки процессоры в состоянии выполнить только несколько десятков команд, которых недостаточно для решения сложных задач. Идентификация точки, видимой в пикселе, требует обработки целой сцены, которая состоит из миллионов объектов.

С другой стороны, вычисление цветов эквивалентно моделированию законов электромагнитных волн и оптики и требует решения интегральных уравнений, ещё усложняющих вычисления. Необходимую вычислительную мощность можно достичь очень распараллеленной вычислительной системой, которая была бы специализирована для решения специфической проблемы визуализации. Эти аппаратные средства называются GPU. Они содержат большое количество конвейерных стадий и производят интенсивные вычисления. Конвейер имеет параллельную архитектуру, несколько тысяч ядер. В результате мощность GPU эквивалента десяткам современных центральных процессоров.

Параллельные вычисления эффективны если взаимозависимость обработки различных элементов данных минимальна. К примеру явление освещения вводит существенную связь между различными элементами. Поверхности могут закрывать друг друга и цвета одних поверхностей могут влиять на цвета других из-за передачи света и отражений. Поэтому основная физическая модель часто упрощается: взаимоотражения игнорируются, что ограничивает связь между

элементами. Это упрощение называют местной (локальной) моделью освещения, так как это предполагает, что точка может быть закрашена, используя только свои собственные свойства. Локальный подход к освещению далёк от точной модели распространения света. Вычислительные схемы, учитывающие взаимозависимости, называются глобальными моделями освещения.

Компьютерная графика делится на две отрасли:

1. Графика реального времени. Использует локальные подходы освещения. Пример: компьютерные игры.
2. Графика, ориентированная на создание фотореалистичных изображений. Часто требует многих часов вычислений.

Существуют языки высокого уровня для программирования GPU. Из них два основных: HLSL (High Level Shader Language) для среды Direct3D и OpenGL.

Виды компьютерной графики

19 сентября 2015 г.
10:55

Различают два вида компьютерной графики:

1. Растровая. Изображение представляется в виде набора окрашенных точек. Используется, к примеру, для создания фотографий или отсканированных иллюстраций. Для ввода растровых изображений используются цифровые фото- и видеокамеры. Примеры: bmp, jpeg, png, tiff. Данные изображения редко создают с помощью компьютерных программ, и большинство графических редакторов ориентированы не столько на создание изображений, сколько на их обработку.
2. Векторная. Представляет изображение в виде совокупности дуг, отрезков многоугольников и т. д. Вектор — это набор данных, характеризующий какой-либо примитив. Используется для создания схем, чертежей, иллюстраций, использующих простейшие геометрические элементы. Примеры: PDF, EPS, SVG.
3. Фрактальная. Программные средства для работы с фрактальной графикой предназначены для автоматической генерации изображения, путём математических расчётов. Создание фрактального рисунка заключается не в рисовании, а в программировании. Является вычислимой так же, как и векторная, но в памяти хранятся не объекты, а только формулы. Изменяя коэффициенты в уравнении можно получать совершенно другой рисунок. Способность фрактальной графики моделировать образы живой природы часто используется для автоматической генерации необычных иллюстраций.

Сравнительная характеристика растровой и векторной графики:

Критерий сравнения	**Растровая**	**Векторная**
Способ представления изображений	Изображение строится из множества пикселей.	Изображение описывается в виде набора примитивов с их параметрами (размеры, координаты).
Представление объектов реального мира	Используется для представления реальных образов.	Не позволяет получать изображения фотографического качества.
Качество редактирования изображения	При масштабировании и вращении растровых картинок возникают искажения.	Изображение преобразовывается без потери качества.

Особенности печати изображения	Легко печатаются.	Иногда не печатаются, а если и печатаются, то не так как нужно.

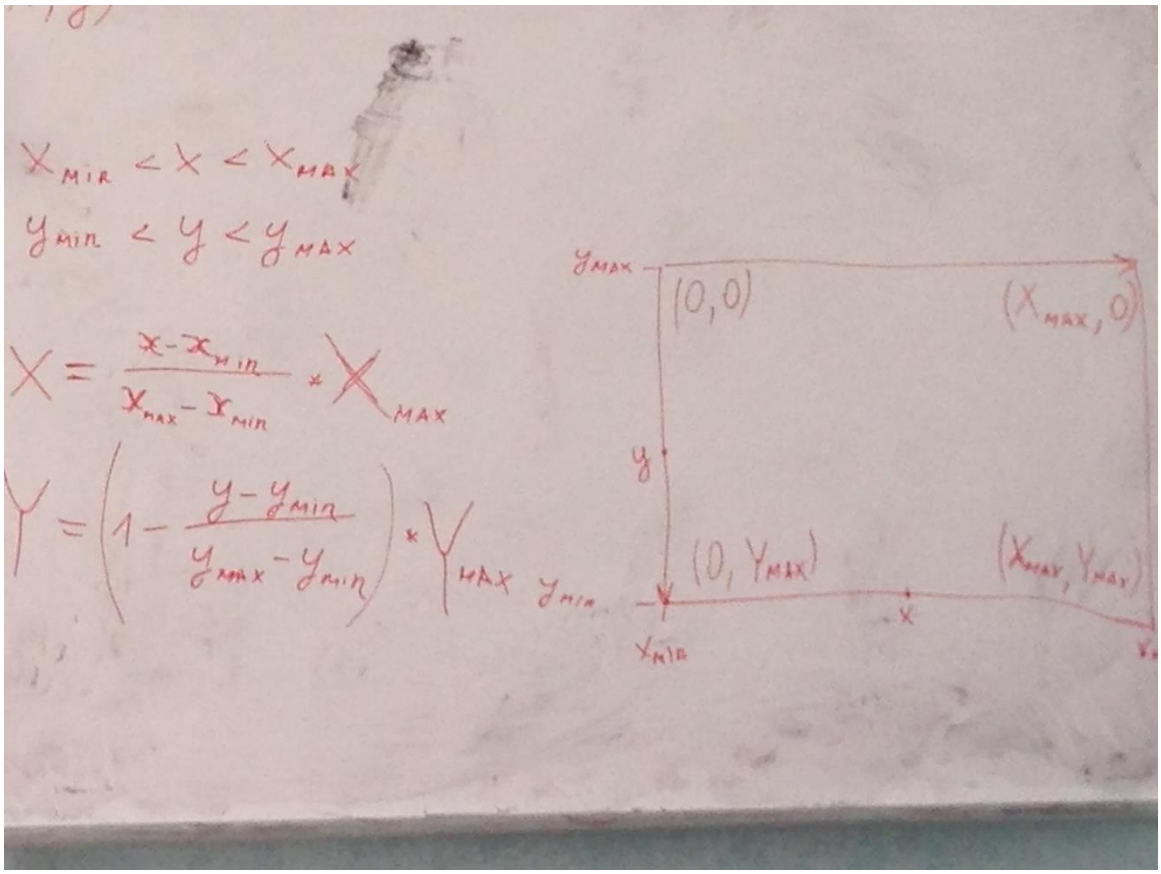
Построение графиков функций

26 сентября 2015 г.
10:03

При работе с компьютерной графикой приходится иметь дело с двумя системами координат.

Первая система — это система координат устройства или экранная система координат. Координатами точки в этой системе является номер пикселя в строке X и номер строки Y. Начало координат расположено в левом верхнем углу. Параметры экранной системы координат: максимальное число пикселей в строке X_{max} и максимальное число строк пикселей Y_{max} — зависят от размера окна.

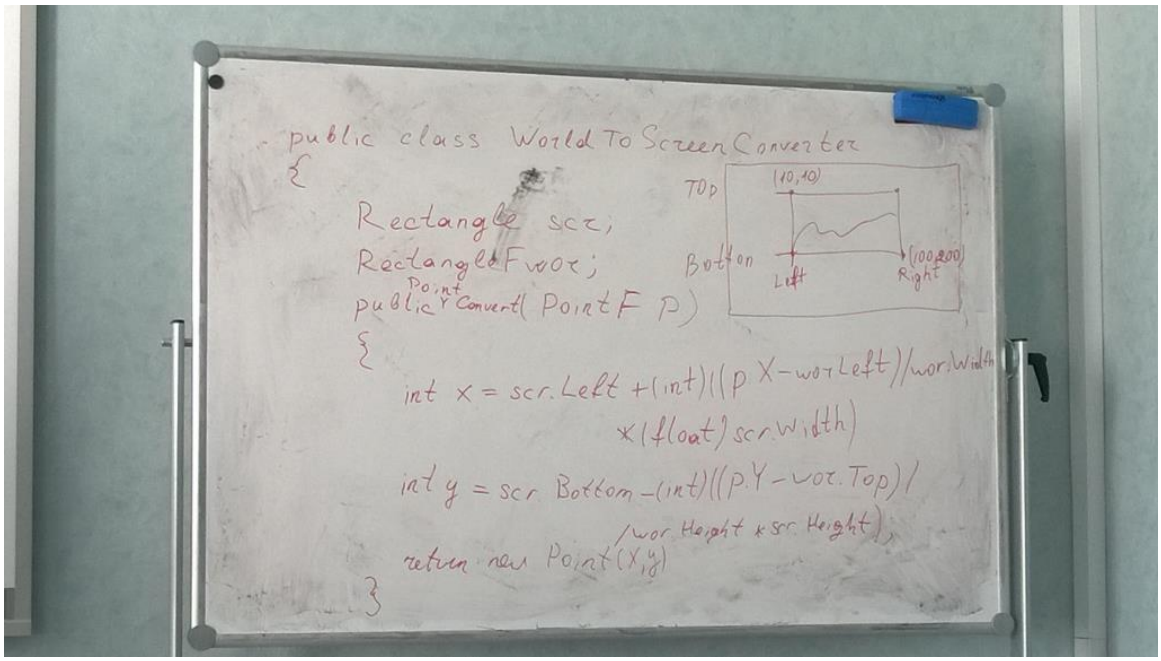
Вторая система координат — это мировая или математическая. Она представляет собой декартову систему XY, определяемую программистом и является независимой от конкретного графического устройства. Параметры, которыми задаются диапазоны изменения X и Y (X_{min}, X_{max}, Y_{min}, Y_{max}) определяют прямоугольную область в математическом двумерном пространстве. Эти параметры зависят от конкретной функции. Мировые координаты и координаты устройства связаны между собой простыми соотношениями.



$$X = \left(\frac{x - x_{min}}{x_{max} - x_{min}}\right) * X_{max}$$

$$Y = \left(1 - \frac{y - y_{min}}{y_{max} - y_{min}}\right) * Y_{max}$$

X, Y — координаты точки в пикселях.



Графики бывают разных видов:

- Точечные
- Линейные
- Графики в виде столбиков
- Графики в виде секторной диаграммы

В качестве задачи построения линейного графика произвольной функции $y = f(x)$, в заданном диапазон
именения аргумента $x \in (a; b)$, с исходными данными $f(x), a, b, h_{\text{предпочт.}}$. $h_{\text{предпочт.}}$ — предпочтительный
сетки, измеряемый в пикселях.

Для разметки осей необходимо использовать параметр $h_{\text{предпочт.}}$. Данный параметр достаточно задать
константой. Шаг сетки в мировой системе координат должен выбираться автоматически из значений $1 * 1$
$10^n, 5 * 10^n, ..$, таким образом, чтобы при пересчёте на пиксели быть наиболее близким к $h_{\text{предпочт.}}$. Значе
$x_i; y_j$ на которых размечаются линии сетки должны нацело делиться на размер шага.

Пример. Предположим, что $x \in [-0.31, 3]$, ширина окна — 800 пикселей, предпочтительный шаг — 50 пи

$\frac{800}{3.31} = 241.69$ пикселей на единицу в мировой системе.

Разметку начинаем с −0.2.

Пример 2. $x \in [12003; 22004]$, ширина окна — 700, предпочтительный шаг 50.

$\frac{700}{10001} = 0.07$ пикселей на единицу в мировой системе. Пробуем: $0.07 * 5 * 10^2 = 35$. Разметку начинаем с
т. к. на 35 пикселей приходится 500 единиц мировой системы.

Чтобы на C# нарисовать лэйблы, нам нужно установить формат вывода числа с помощью.

При ресайзе должны пересчитываться лэйблы на осях. Необязательное требование: писать на форме функцию.

Аффинные преобразования на плоскости

3 октября 2015 г.
10:05

Допустим на плоскости введена прямолинейная координатная система и дана точка $M(x, y)$. Сопоставим т
вторую точку M' с координатами (x', y'). Координаты точек связаны следующими соотношениями (*):

$$x' = \alpha x + \beta y + \lambda$$
$$y' = \gamma x + \delta y + \mu$$

Где: $\begin{vmatrix} \alpha & \beta \\ \gamma & \delta \end{vmatrix} \neq 0$. Можно понимать, что в процессе данного преобразования точка M перемещается в точку

Рассмотрим следующие простейшие геометрические операции:

Поворот вокруг точки (0, 0) на угол ϕ. Он описывается следующими формулами:

1. $x' = x\cos\phi - y\sin\phi$
   $y' = x\sin\phi + y\cos\phi$

Растяжение или сжатие вдоль координатных осей:

2. $x' = ax$
   $y' = by$

Отражение (например, относительно оси X):

3. $x' = x$
   $y' = -y$

Перенос:

4. $x' = x + \lambda$
   $y' = y + \mu$

В курсе аналитической геометрии доказывается, что любое преобразование вида (*) можно представить, как последовательное исполнение простейших преобразований вида (1)–(4). Обычно используется матричная запись данных преобразований.

Для первого случая матрица имеет вид:

$$(x', y') = (x, y)\begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix}$$

Для второго:

$$(x', y') = (x, y)\begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}$$

Для третьего:

$$(x', y') = (x, y)\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Необходимо охватить матричным способом все четыре преобразования (в том числе и перенос). Это достигается следующим образом. Необходимо перейти к описанию произвольной точки плоскости не упорядоченной парой чисел, а тройкой чисел.

Пусть дано $M(x, y)$. Однородными координатами точки M называется любая тройка одновременно нерав
чисел x_1, x_2, x_3, связанных следующими соотношениями:

$$\frac{x_1}{x_3} = x, \frac{x_2}{x_3} = y$$

При решении задач компьютерной графики однородные координаты обычно вводят так: $M(x, y) \rightarrow M(x, y$
(точке M ставится в соответствие точка M с изменёнными координатами).

При помощи троек однородных координат и матриц третьего порядка можно описать любое аффинное преобразование на плоскости, в том числе и перенос. Таким образом выражению (*) соответствует матрич

запись $(x', y', 1) = (x, y, 1)\begin{pmatrix} \alpha & \gamma & 0 \\ \beta & \delta & 0 \\ \lambda & \mu & 1 \end{pmatrix}$

Возможна запись выражения по столбцам.

Элементы произвольной матрицы аффинного преобразования не несут явного смысла. Поэтому чтобы реализовать то или иное отображение необходимы специальные приёмы. Для того, чтобы выполнить последовательность преобразований необходимо перемножать матрицы. Выпишем матрицы третьего порядка для основных геометрических операций:

1. Поворот: $[R] = \begin{pmatrix} cos\phi & sin\phi & 0 \\ -sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$
2. Растяжение и сжатие: $[D] = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{pmatrix}$
3. Отражение: $[M] = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
4. Перенос: $[T] = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \lambda & \mu & 1 \end{pmatrix}$

Рассмотрим пример. Пусть дана точка $A(x_a, y_a)$ и произвольный четырёхугольник, который необходимо повернуть вокруг этой точки на угол ϕ.

В таком случае, сперва применим перенос:

$[T_1] = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_a & -y_a & 1 \end{pmatrix}$

Далее поворачиваем (умножая $[T_1]$ на $[R]$):

$[R] = \begin{pmatrix} cos\phi & sin\phi & 0 \\ -sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$

И переносим обратно (умножая ещё и на $[T_2]$):

$[T_2] = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_a & y_a & 1 \end{pmatrix}$

Итог: $[T_1] * [R] * [T_2]$

Таким образом:

$(x_i, y_i)M_1 = (x_i', y_i')$

Аффинные преобразования в пространстве

6 декабря 2015 г.
16:05

Рассмотрим трёхмерный случай и сразу введём однородные координаты. Заменим координатную тройку (x, y, z), задающую точку в пространстве на четвёрку чисел $(x, y, z, 1)$. Данный переход к новому способу з точек даёт возможность воспользоваться матричной записью и в более сложных трехмерных задачах. Люб аффинное преобразование в трёхмерном пространстве может быть представлено в виде комбинации вра растяжений, отражений и переносов.

Матрица вращений вокруг оси абсцисс на угол ϕ:

$$[R_x] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\phi & sin\phi & 0 \\ 0 & -sin\phi & cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Матрица вращения вокруг оси ординат на угол ψ:

$$[R_y] = \begin{pmatrix} cos\psi & 0 & -sin\psi & 0 \\ 0 & 1 & 0 & 0 \\ sin\psi & 0 & cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Матрица вращения вокруг оси аппликат на угол χ:

$$[R_z] = \begin{pmatrix} cos\chi & sin\chi & 0 & 0 \\ -sin\chi & cos\chi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Матрица сжатия/растяжения (масштабирование):

$$[D] = \begin{pmatrix} \alpha & 0 & 0 & 0 \\ 0 & \beta & 0 & 0 \\ 0 & 0 & \gamma & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \alpha, \beta, \gamma > 0$$

α — коэффициент сжатия относительно оси X (остальные аналогично Y и Z).

Матрица отражения относительно плоскости XY:

$$[M_z] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Матрица отражения относительно плоскости YZ:

$$[M_x] = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Матрица отражения относительно плоскости XZ:

$$[M_y] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Матрица переноса:

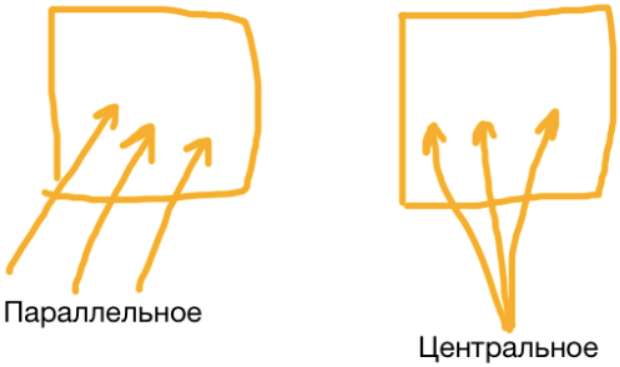
$$[T] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a & b & c & 1 \end{pmatrix}$$

a, b, c — переносы, соответственно, по осям X, Y, Z.

Виды проекций

10 октября 2015 г.
11:00

Изображение объектов на экране связано с ещё одной геометрической операцией — проектированием при помощи пучка прямых. В компьютерной графике используется несколько различных видов проектирования. Чаще всего применяются параллельное и центральное.



Параллельные:

1. Ортографическая — пучок проецирует объект в реальных размерах.
2. Аксонометрическая (триметрическая, диметрическая и изометрическая) — как ортографическая, но объект под разными углами.
3. Косоугольная — пучок падает на плоскость под углом.
    a. Свободная
    b. Кабинетная

Перспективные:

1. Одноточечная
2. Двухточечная
3. Трёхточечная (центральная) — пучок выходит из одной точки. При бесконечном удалении точки от объекта вырождается в ортографическую.

Матрицы проекций

Ортографическая проекция на плоскости YZ:

$[P_x] = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$, то есть при умножении любой строки на эту матрицу, координата X обнулится. На д плоскости аналогично.

Косоугольная проекция — это проекция для получения которой используется пучок прямых неперпендикулярных плоскости.

Матрица косоугольного преобразования на плоскости XY:

$[K] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \alpha & \beta & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

В случае свободной проекции $\alpha = \beta = \cos\frac{\pi}{4}$, а в случае кабинетной $\alpha = \beta = \frac{\cos\frac{\pi}{4}}{2}$.

Перспективные проекции строятся следующим образом. Предположим, что центр проектирования лежит Z в точке $C(0, 0, Z_c)$. Тогда матрица имеет вид:

$$[C] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{Z_c} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Применяются также трёхточечные проекции, и матрица принимает следующий вид:

$$[C] = \begin{pmatrix} 1 & 0 & 0 & -\frac{1}{a} \\ 0 & 1 & 0 & -\frac{1}{b} \\ 0 & 0 & 1 & -\frac{1}{c} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Обычно умножается матрица на строку. В нашем случае нужно умножать строку на матрицу, если же нужно наоборот, то следует сперва транспонировать матрицу.

Современные 3D API

17 октября 2015 г.
10:04

В современных 3D-ускорителях содержится больше транзисторов, чем в центральных процессорах. У каждой видеокарты есть свой Ассемблер и специальные команды, через которые ей можно управлять. Трёхмерные программные интерфейсы приложений выполняют следующие функции:

1. Взаимосвязь с ресурсами центрального процессора и памятью.
2. Взаимосвязь с драйвером графического процессора.
3. Взаимосвязь с интерфейсом операционной системы.
4. Предоставление библиотечных функций для создания трёхмерной сцены.
5. Обеспечение через функции библиотеки полного прохождения конвейера обработки сцены.

OpenGL

OpenGL (Open Graphic Library) — создана в 1992 году корпорацией Silicon Graphics. OpenGL — это универсальная аппаратно-независимая библиотека, которая поддерживает разнообразные 3D-объекты и конструкции, начиная с примитивов типа треугольника или линии и заканчивая сложными поверхностями. OpenGL — это всего лишь спецификация и все детали её реализации возлагаются на производителей аппаратного обеспечения.

В OpenGL есть механизм расширений, позволяющий добавлять в библиотеку какие-то функции, нереализованные базовой версией API. Начиная с версии 2.0 OpenGL получил встроенное дополнение под названием OpenGL Shader Language. Что позволило программистам выбирать между использованием жёстко прописанных функций и программируемыми шейдерами.

Важной особенностью OpenGL является кроссплатформенность. Она доступна на многих ОС, поэтому ядро OpenGL не поддерживает никаких функций для обработки окон.

Direct 3D

Direct 3D — это элемент комплексной библиотеки DirectX, разработанной компанией Microsoft для расширения и универсализации мультимедийной возможности операционной системы Windows.

Direct 3D работает в двух режимах:

- Retained Mode (режим абстракции) — библиотека общается с аппаратным обеспечением напрямую и даёт наибольшую производительность.
- Immediate Mode

Также существует режим HEL (Hardware Emulation Layer) при котором используются только ресурсы ЦП. Качество графики при этом явно плохое.

Начиная с версии 8.0 добавлена поддержка шейдерных операций.

Для Direct 3D существует язык HLSL.

Direct 3D популярен у разработчиков компьютерных игр

Glide

Является урезанной версией OpenGL, использует его базовые функции обработки геометрии, но является закрытым и ориентирован только на архитектуры VooDoo. Использовался для создания игр.

OpenGL

24 октября 2015 г.
10:11

Трехмерная графика использует эту идею и представляет только внешнюю оболочку объекта, вместо того, чтобы представить его целиком.

В трехмерной графике все объекты представляются набором полигонов (многоугольников) соединенных в форме объекта. Каждую конечность полигона называют вершиной (vertex). Вершины — это ядро трёхмерной графики, поскольку они несут всю необходимую для представления информацию, такую как местоположение, цвет и данные текстуры. На вход поступают данные, описывающие трёхмерную сцену, а выходом является двухмерная растровая картинка.

Объект выводится в несколько этапов. Первый из них — триангуляция. Триангуляция — разбиение на треугольники. Треугольник является простейшим полигоном, вершины которого однозначно задают плоскость (потому что через 3 точки всегда можно провести плоскость; с четырёхугольниками такое не прокатит).

Любую область или поверхность можно гарантированно разбить на треугольники. Вычислительная сложность алгоритмов разбиения на треугольники существенно меньше, чем при использовании других полигонов.

Для треугольника легко определить три его ближайших соседа, имеющих с ним общие грани.
В OpenGL вывод треугольника на экран делается следующим образом:

```
glBegin(GL_TRIANGLE);
/* операторные скобки + константа вывода треугольника (т. е. дальше должны выводиться
координаты/параметры трёх вершин */
        glVertex3f(10.0f, 5.0f, 3.0f);
        /* 3 — трёхмерная графика, f — параметры типа float */
        glVertex3f(..);
        glVertex3f(..);
glEnd();
```

У вершины могут быть дополнительные свойства, помимо её местоположения. Например, вершине может быть назначен цвет или с ней может быть связана нормаль. Direct 3D и OpenGL обладают значительной гибкостью и позволяют конструировать собственные форматы вершин. Другими словами, они позволяют нам указать, какая информация будет содержаться в данных вершинах. Для задания цвета используется функция glColor:

```
glColor3f(1.0f, 0.0f, 0.0f)
/* цифры задают rgb [0..1] */
```

Чтобы создать собственный формат вершин, нам сначала необходимо создать структуру, которая будет хранить необходимые нам данные вершины. Предположим, нам нужно создать структуру, которая хранит координаты и цвет:

```
struct ColorVertex {
        float _x, _y, _z;
        DWORD _color;
}
```

Рассмотрим пример рисования квадрата на плоскости x = 1, с указанием вектора нормали. Вектор имеет координаты (1; 0; 0):

```
glBegin(GL_QUADS);
        glNormal3f(1.0f, .0f, .0f);
        glVertex3f(1.0f, .0f, .0f);
        glVertex3f(1.0f, .0f, 1.0f);
        glVertex3f(1.0f, 1.0f, 1.0f);
        glVertex3f(1.0f, 1.0f, .0f);
glEnd();
```

В некоторых случаях, если накладывается текстура на объект, то необходимо указывать её координаты.

Понятие индексов

Часто образующие трёхмерный объект треугольники имеют общие вершины. Для того, чтобы не хранить копии вершин, используются индексы. Создаётся список вершин и список индексов. В списке вершин перечисляются все уникальные вершины, а список индексов содержит последовательность номеров вершин из списка вершин.

```
Vertex vertexList[4] = { v0, v1, v2, v3 };
int indexList[6] = { 0, 1, 2, /* треугольник №1 */
                     0, 2, 3 /* №2 */ }
```

Трансформация объектов

Во время трансформации преобразуются координаты объектов. К объектам применяется матрица преобразований.

Поворот:

```
glRotatef(angle, x, y, z); /* x, y, z — вектор, вокруг которого осуществляется поворот; угол в градусах */
```

Перенос:

```
glTranslatef(dx, dy, dz);
```

Масштабирование:

```
glScale(sx, sy, sz);
```

Сперва нужно вызывать функции трасформации, а потом уже функции для вывода.

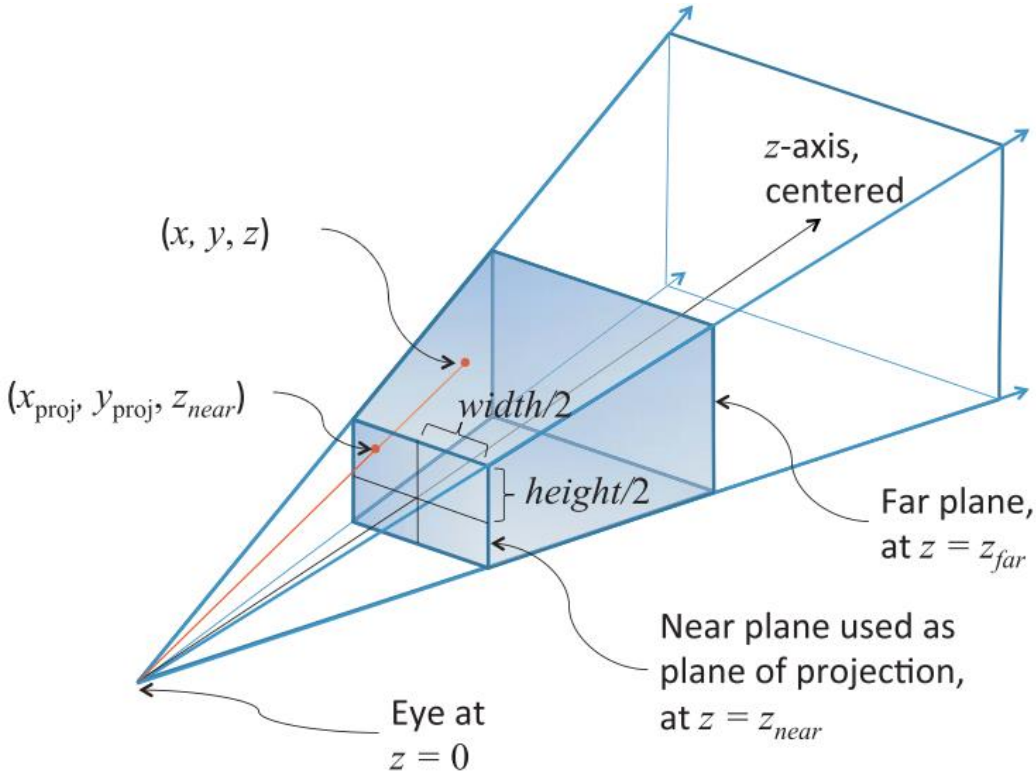
Пример: Объект с центром в (cx, cy, cz) требуется перенести и повернуть его на угол angleX и angleY и вывести его на оси Z.

```
glLoadIdentity(); /* загружает единичную матрицу */
glTranslatef(0, 0, zoffset);
glRotate(anglex, 1.0f, .0f, .0f);
glRotatef(angley, .0f, 1.0f, .0f);
glTranslatef(-cx, -cy, -cz);
DrawObject();
```

Сцена

6 декабря 2015 г.
16:20

Преобразование вида заключается в том, что координаты объектов, занимавших места в системе координат сцены, транслируется в координаты, привязанные к виртуальной камере. Это делается для упрощения следующего шага конвейера. Проецирование в результате этого преобразования камера перемещается в начало координат и поворачивается так, чтобы быть направленной вдоль положительного направления оси Z. Все объекты сцены так же подвергаются этому преобразованию так, что формируемый камерой вид сцены не изменяется. Камера определяет какую часть мира видит зритель и, следовательно, для какой части мира нам надо создавать. Камера позиционируется и ориентируется в пространстве и определяет видимую область пространства. Область видимого пространства представляет собой усеченную пирамиду (frustum) и определяется углами поля зрения передними и задними плоскостями.



Объекты, которые не находятся между передней и задней плоскостью отсечения не выводятся.

glFrustum(vLeft, vRight, vBottom, vTop, vNear, vFar)
vLeft, vRight – координаты плоскостей отсечения слева и справа
vBottom, vTop – координаты плоскостей отсечения снизу и сверху
vNear, vFar – координаты ближней и дальней плоскостей

glFrustum(-1, 1,-1, 1, 2, 11);

Необходимо, чтобы передняя и задняя плоскость отсечения были как можно ближе друг к другу. В этом случае вычислений приходится делать меньше.

Для вывода ортографической проекции используется функция glOrto с такими же параметрами.

glPerspective(angle, width/height, vNear, vFar)
angle – угол
vNear, vFar – координаты ближней и дальней плоскостей

glLookAt — задает координаты позиции наблюдателя, координаты контрольной точки находящейся в центре экрана и направление камеры.

Следующий этап конвейера — проекция сцены.
На этом этапе происходит проецирование трехмерных объектов сцены на двух мерную плоскость. То, что видно на этой плоскости является результатом съемки виртуальной камеры. Эта двухмерная область, на которую проецируется находящиеся внутри области видимого пространства трехмерные объекты для создания двухмерного изображения. Direct 3D и OpenGL определяют плоскость проекции как плоскость z = 1.

Отсечение невидимых частей.

Для того, чтобы на вход следующих этапов конвейера не поступало лишней информации, т.е. объекты или фрагменты объектов, которые не попадают в поле видимости камеры применяются различные методы отсечения невидимых частей, определяются полигоны, которые гарантированно не попадут в кадр, эти полигоны отсекаются. Требуется так же учесть полигоны, у которых в кадре только кусок. Так же не нужно выводить полигоны объекта, которые заслонены другими объектами. Для отсечения сначала используются объем отсечения (6 плоскостей по 3 координатам, которые ограничивают область сцены). Затем необходимо отбросить задние грани. У каждого полигона есть две стороны лицевая и обратная, а также важнейшая характеристика – нормаль. К примеру сфера образована большим количеством полигонов, нормали полигонов сферы направленны во внешнюю сторону и ее поверхность образована лицевыми сторонами граней. Примерно половина полигонов сферы направлена от экрана(нормали), вторая половина на экран. Это значит, что все полигоны направлены от экрана, гарантированно не попадут в кадр.

Растеризация

7 ноября 2015 г.
10:09

Растеризация — это непрограммируемый этап. При растеризации 3D-картинка, спроецированная на плоскость, преобразуется в растровый формат, то есть определяются правильные значения результирующих пикселей. Для каждого пикселя определена глубина, и она заносится в z-буфер.

Глубина — это расстояние от экрана до объекта, а z-буфер — это буфер, в котором накапливается информация о глубине каждого объекта.

Альясинг (aliasing — помеха дискретизации) возникает из-за несоответствия изображения реального объекта и его представления в компьютере. В реальном мире объекты имеют сглаженные линии и контуры, монитор же может отображать лишь дискретные точки, то есть пиксели. Так как пиксели имеют форму квадратов и равномерно закрашиваются определённым цветом, линии объектов становятся зубчатыми или, иначе говоря, ступенчатыми. Для борьбы с этой проблемой существует простейший метод — множественные выборки (multisampling).

Каждый пиксель разбивается на субпиксели. Для дополнительной информации о пикселе выделяется дополнительный буфер. При применении такого сглаживания фактически получается так, что изображение рассчитывается несколько раз, что резко уменьшает производительность, но зато получаются более гладкие переходы цвета соседних пикселей на линиях объектов.

Вся сцена обрабатывается в каком-то большом виртуальном разрешении, а затем сжимается до фактического разрешения. В общем случае виртуальное и фактическое разрешения могут быть некратными. Такая техника называется supersampling.

Обработка пикселей

На этой стадии отдельным пикселям присваиваются значения цвета, которые были получены интерполяцией цветов вершин, либо они заменяются или к ним прибавляется цвет текстуры. Также здесь может действовать пиксельный шейдер, который будет определённым образом комбинировать свет, глубину и позицию пикселя с текстурами по специальному алгоритму.

Шейдер — подпрограмма, выполняемая или аппаратно, или на графическом чипе. Это могут быть подпрограммы, которые векторно обрабатывают массивы объектов. Современные видеокарты поддерживают векторные вычисления, поэтому код, выполняемый на них, может работать быстрее, чем на центральном процессоре.

После прохождения всех стадий конвейера картинка заносится в буфер кадра. У буфера кадра имеется два или более слоя — передний и задний. В заднем — новый кадр, а в переднем — текущий. Когда приходит время очередного цикла обработки сцены, содержимое этих слоёв, меняется местами. В итоге на экране мы видим новый кадр, а старый кадр посылается в конец буфера, где немедленно заменяется очередным, только что просчитанным.

Текстурирование

7 ноября 2015 г.
10:39

Благодаря процессу текстурирования, объекты получают реалистичность. Текстура — это двумерная битовая карта (картинка), которая накладывается на полигон и изображает его поверхность. То, что нельзя смоделировать полигонами, можно просто нарисовать текстурой. Если на поверхности нет сильно выдающихся или неоднородных деталей, оно будет смотреться реалистично, если на неё наложить текстуру.

По аналогии с конечным изображением, которое состоит из пикселей, текстура состоит из т. н. текстелей. Тексели — это пиксели текстуры, как растровые картинки. Текстура накладывается строго по координатам. Процесс наложения текстур алгоритмически выведен для объектов типа сферы, цилиндра, конуса и многоугольника.

При работе с текстурами существует немало проблем. У экрана есть своё разрешение и определённое количество пикселей, которые на нём можно отобразить. Но в соответствующей области виртуального мира количество текселей может быть и больше, и меньше, чем на экране. Отсюда появляются проблемы отображения текстурированных объектов. Для борьбы с ними используются различные методы фильтрации:

1. Билинейная. Её суть в том, что цвет пикселя получается в результате интерполяции (усреднения) цветов четырёх соседних текселей. Если объект расположен далеко от камеры, его текстура почти не искажается, а когда объект недалеко от камеры, и текселей не хватает, интерполяция создаёт расплывчатое изображение этой области. Билинейная фильтрация хорошо работает только для полигонов, которые параллельны или почти параллельны экрану. Дело в том, что четыре соседних текселя, которые берутся для интерполяции, —это почти круг. Если плоскость наклоняется, то проекция круга превращается в эллипс, но интерполируются тексели по-прежнему по кругу. От этого постоянно накапливаются небольшие ошибки. После определённого угла наклона ошибки становятся уже заметными, текстура фильтруется геометрически неправильно и пользователь наблюдает сильные искажения.

   Если мы видим текстуру намного дальше или намного ближе, чем предполагал разработчик, текстура очень сильно искажается фильтрацией. До поры, до времени это незаметно, но со временем в рисунке появляются помехи. Решение данной проблемы — это трилинейная фильтрация.
2. Трилинейная (MIP Mapping). Для одной и той же поверхности, разработчики создают несколько копий текстуры с разной степенью детализации. Каждая следующая версия текстуры меньше или больше предыдущей в четыре раза. Текущая текстура называется MIP-уровнем, а все MIP-уровни вместе — MIP-каскадом. Когда камера удаляется от текстуры, она заменяется на MIP-уровень с меньшим разрешением. А когда приближается — с большим. Теперь независимо от того, на каком расстоянии находится наблюдатель, текстура отображается без геометрических искажений. Однако в памяти приходится хранить несколько копий одной и той же текстуры. Переходы между текстурами происходят резко.

   Трилинейная фильтрация объединяет MIP Mapping и билинейную фильтрацию. Цвет

конкретного пикселя на экране определяется в результате интерполяции цветов текселей двух соседних MIP-уровней. При этом над каждым MIP-уровнем предварительно проводится билинейная фильтрация. А для интерполяции берутся 4 или 8 соседних текселей. За счёт этого переход между MIP-уровнями становится плавным и незаметным.

3. **Тут начинается лекция за 14.11.15.** Анизотропная. Самая сложная и ресурсоёмкая из всех. После всех фильтраций и MIP Mapping остаётся одна проблема — более или менее красиво эти алгоритмы работают для текстур, которые располагаются параллельно экрану. Если текстура сильно наклонена, фильтрация в одном направлении выполняется больше, чем надо, а в другом меньше. Поэтому и появилась анизотропная фильтрация.
   Анизотропная фильтрация оперирует не менее, чем восемью текселями, во все стороны MIP-уровней, при этом используются модель с заранее неопределённой формой. В результате убираются шумы и искажения объектов, и изображение получается более качественным. Анизотропная фильтрация работает с текселями как с эллипсами, и для получения одного пикселя обрабатывает до 32 текселей.
   Качество текстуры при анизотропной фильтрации даже на дальних дистанциях остаётся схожей с оригинальной; при изотропной фильтрации изображение сглаживается, поэтому теряется качество.

Процедурные текстуры

Процедурные текстуры — это текстуры, описываемые математическими формулами. Примеры текстур: дерево, вода, облака, кристаллическая или плиточная поверхность, ткань и т. д.
Преимущество процедурных текстур следующие:

1. Они не занимают в видеопамяти место, т. к. просчитываются в реальном времени.
2. Позволяют использовать анимации при помощи всего лишь небольшой модификации математических формул.
3. Включают в себя неограниченный уровень детализации каждой текстуры. Поэтому пикселизации не будет. Текстура всегда генерируется под необходимый для её отображения размер. Данные текстуры могут неограниченно масштабироваться в реальном времени и для них не требуются MIP-уровни.

Процедурными текстурами можно имитировать любой однородный материал. Наиболее распространённые алгоритмы для генерации таких текстур — вариации «шумовых» алгоритмов Перлина. Недостаток процедурных текстур — это их естественное требование к большому количеству вычислений, которых существенно больше, чем при загрузке обычной растровой текстуры в память. Первоначально подобные текстуры считались программно центральным процессором. В дальнейшем появились расширения процессора MMX, 3D NOW, SSE, AVS512, ориентированные на векторную обработку данных. Наложение процедурных текстур стало происходить с такой же скоростью, как и обычных.

Наложение текстур на объект

Текстура накладывается на определённые участки каркаса объекта. Карта текстуры — это двумерный рисунок, назначаемый плоской поверхности, т. е. это растровое изображение, которое повторяется, если его копировать и вставить рядом. Проекционные координаты — это карта поверхности трёхмерного объекта, необходимые для корректного размещения на них текстуры. Проекционные координаты также называют UV-координатами. Они указывают как размещать двумерный рисунок, на поверхности модели. Для полигональных поверхностей обычно используются проекционные координаты следующих типов:

1. Плоские (Planar)
2. Цилиндрические (Cylindrical)
3. Сферические (Spherical)

Кроме того, используется особый метод, называемый автоматическим проецированием. Применение плоских проекционных координат приводит к размыванию рисунка в областях, перпендикулярных направлению проецирования. Этот метод часто применяется для больших

объектов, когда зритель видит только одну грань объекта. Плоское наложение является самым простым в использовании, поскольку требует только задания в направлении наложения текстуры на объект. Т. к. при плоском наложении текстура накладывается на объект только в одном направлении, боковые грани объекта получаются полосатыми. Цилиндрические координаты используются для наложения текстур на объекты в форме цилиндров. Аналогично, сферические — для сфер.

Освещение

21 ноября 2015 г.
10:07

Видимый свет — это ЭМИ с диапазоном волн от 380 до 780 нм и с частотой от 750 ТГц до 400 ТГц.

Красивое изображение получается во многом благодаря удачному выбору освещения. Лучи света от виртуального источника могут проходить сквозь какой-нибудь объект, освещая расположенные за ним объекты, которые в реальности оставались бы в тени. Кроме того, в компьютерной графике объекты не рассеивают падающий на них свет, поэтому освещёнными оказываются только области, непосредственно расположенные на пути лучей света.

В современных программах для создания трёхмерной графики и анимации можно смоделировать такие экзотические ситуации, как отрицательное освещение, когда источник света не увеличивает, а уменьшает освещённость сцены, и включение объектов в освещение, когда источником освещаются только специально указанные объекты. Освещение может приводить к появлению теней, а также изменять вид материалов и раскраски объектов.

Существует несколько различных типов источников света:
1. Направленный источник (directional) (пример: солнце)
2. Рассеянный (ambient) — освещает все объекты сцены, независимо от их положения.
3. Точечный (point) — испускает лучи из одной точки, равномерно во всех направлениях.
4. Прожектор (spot) — это источник, лучи которого расходятся коническим пучком из одной точки.
5. Прямоугольный (area) — испускает лучи из прямоугольной области.

Свет, уходящий с поверхности в конкретной точке в сторону наблюдателя, представляет собой сумму компонентов, умноженных на коэффициент, связанный с материалом и цветом поверхности. К таковым компонентам относятся:
1. Свет, пришедший с обратной стороны поверхности, т. е. преломленный свет (refracted).
2. Свет, рассеиваемый поверхностью (diffuse)
3. Зеркально отражённый свет (reflected)
4. Блики, т. е. отражённый свет источника (specular)
5. Собственное свечение поверхности (self-illumination)

Наличие теней помогает яснее определить размер объекта, его положение и ориентацию в пространстве. Соответственно формирование теней позволяет определить пространственные соотношения между объектами. Отсутствие теней в сцене приводит к появлению в результате визуализации плоской картины. То есть тени добавляют сцене глубину и делают её более реальной. Кроме задания освещения необходимо задавать и другие физические параметры (гравитацию, свойства атмосферы и т. д.) и свойства взаимодействующих объектов поверхностей. Без настройки этих параметров невозможно получить хорошую модель окружающей среды.

Модели освещения OpenGL и Direct3D свет, испускаемый источниками, состоит из трёх составляющих:

1. Фоновый или рассеянный свет (ambient light) — этот тип освещения моделирует свет, который отражается от других поверхностей и освещает всю сцену.
2. Рассеиваемый свет (diffuse light) — этот свет распространяется в одном направлении, сталкиваясь с поверхностью он отражается равномерно во всех направлениях. Поэтому интенсивность света, который достигает глаза зрителя не зависит от точки с которой просматривается сцена, и местоположение зрителя можно не учитывать. При вычислении рассеянного освещения надо учитывать только направление световых лучей и позицию поверхностей.
3. Отражаемый свет (specular light) — этот свет распространяется в заданном направлении, сталкиваясь с поверхностью он отражается в одном направлении, формируя блики, которые видимы под определённым углом. Отражаемый свет используется для моделирования света от освещённых объектов, которые создают блики, появляющиеся при освещении полированных поверхностей.

Цвет объектов, которые мы видим, определяется цветом отражаемого ими света. Например, красный шар выглядит красным потому, что он поглощает весь диапазон волн, кроме красных. Поэтому красный цвет отражается от шара и попадает в глаз зрителя. В компьютерной графике данное явление моделируется путём задания материала объекта. Материал позволяет задать отражающую способность коллекции.

В современных системах понятие материала расширяется добавлением текстур, моделирующих вид самой поверхности, и подпрограмм шейдеров, определяющих, каким образом будут рассчитываться конечные цвета по описанным выше параметрам материала и источникам света.

Графические протоколы

6 декабря 2015 г.
16:27

Аппаратно-зависимые графические протоколы

Разрабатываются производителями графического оборудования. Представляют собой последовательность команд для построения изображения на устройствах выпускаемых конкретной фирмой. Для интерпретации таких протоколов не требуется дополнительных ресурсов. Первый тип, Tictronic -- разрабатывается производителями графических дисплеев, предусматривает следующие основные группы графических команд:

1. Построение векторных примитивов
2. Работа с растровой графикой
3. Управление сегментами изображений
4. Задание цветовых и геометрических атрибутов
5. Графический ввод
6. Управление плоскостями вывода
7. Выполнение видовых преобразований
8. Определение символов графических объектов

Протокол Regis

По функциональным возможностям данный протокол слабее протокола Tectronic в нем меньше функций растровых операций, функций заданий атрибутов построения и средств графического вывода.

Протокол HPGL (HP Graphic Language)

Разработан фирмой HP в 19.6 году, поддерживается множеством фирм выпускающих плоттеры, принтеры, графопостроители. Всего в языке больше 80 операторов.

Языки описания страниц
Это язык для описания текста и графики на высоком уровне в терминах абстрактных графических элементов. Выполнение вывода происходит в два этапа:

1. Приложение генерирует аппаратно-независимые описания на языке описания страниц
2. Программа управляющая некоторым растровым устройством вывода интерпретирует описание и отображает его на устройство

PostScript
Разработан фирмой Adobe и используется в качестве высокоуровневого аппаратного протокола обмена между компонующей и отображающей системой. Это в первую очередь язык программирования со встроенными мощными графическими примитивами. С другой стороны это язык описания страниц включающий в себя возможности языка программирования использующийся для связи с печатающими устройствами. Для построения изображения в данном языке предоставлена возможность управления каждой точкой печатающего устройства. Аппаратная независимость достигается за счет того, что построение ведется в пользовательской системе координат с помощью обычных графических примитивов и изображение не содержит информацию об устройстве отображения.

Поддерживаемые типы данных:

1. Числа
2. Строки
3. Одномерные массивы
4. Словари
5. Таблицы

Примитивы управления включают условия циклы и процедуры.

В язык встроены следующие возможности:

1. Построение изображений из отрезков, дуг, кубических кривых и других графических примитивов
2. Примитивы могут быть выведены линиями требуемого вида, закрашены любым цветом или использованы для задания области отсечения
3. Текст полностью интегрирован с графикой, т.е. символ интерпретируется, как графические образы.
4. Двумерная координатная система поддается линейным преобразованиям (сдвиг, поворот, масштабирование)
5. Все изображения могут иметь требуемое разрешение

Язык PostScript используется как язык для описания страниц для лазерных принтеров с большим разрешением.

PCL
Вывод на лазерные принтеры рисунков и текстов с использованием различных шрифтов. Символьное кодирование используемое в PCL значительно более компактное и обрабатывается быстрее чем в языке PostScript.

Аппаратно-независимые графические протоколы (метафайлы)
Представляют собой процедуру описания изображений. Метафайлы обеспечивают возможность запоминать графическую информацию в одном месте, передавать ее между различными

графическими элементами и интерпретировать информацию для вывода на графические устройства.

NAPLPS
Это американский стандарт на представление графических данных в сетях.

Возможности протокола:
1. Передача графической информации в потоке символьных данных
2. Минимальный объем передаваемых данных
3. Для интерпретации и вывода изображений требуется минимум усилий

Информация кодируется 7 или 8 битными ASCI кодами.
Недостаток: невозможно получить изображение высокого качества.

GKSM
Содержит небольшой объем графических примитивов, позволяет кодировать изображение в текстовом формате, что позволяет читать и редактировать метафайл.

CGM
Это стандартные графические метафайлы, которые могут совмещаться с различными программными системами.
Использует три способа кодирования: символьное, двоичное и текстовое.

WMF
Использует только двоичный способ кодирования. Метафайл содержит заголовок и описание изображения в виде набора GDI функций.

Символьное кодирование предназначено для передачи и хранения графической информации.
Двоичное кодирование удобно при генерации и интерпретирования изображения.
Текстовое кодирование наиболее наглядное и позволяет редактировать метафайлы.

Растровые и графические файлы

BMP (Bit Map)
Изначально формат растровой графики Windows. В котором эта система хранит свои растровые массивы используются расширения .bmp (информация о цвете каждого пикселя кодируется 1,4,8,16,24 битами на пиксель, числом бит на пиксель называется глубина представления цвета, которая определяет максимальное количество цветов в изображении) или .rle (означает, что произведено сжатие растровой информации)

Файл BMP разбивается на 4 основных раздела:
1. Заголовок файла растровой графики
   Содержит информацию о файле
2. Информационный заголовок растрового массива
   Содержатся сведения об изображении (ширина, высота, метод сжатия, число цветов и другие параметры)
3. Таблица цветов
   Представлены значения основных цветов в формате RGB для используемых в изображении цветов
4. Данные растрового массива
   Формат данных растрового массива в файле BMP зависит от числа бит используемых для кодирования данных каждого пикселя. При 256 цветном изображении, каждый пиксель в

той части файла, где содержатся данные растрового массива описывается 1 байтом. Это описание не представляет значение цветов RGB, а служит указателем для входа в таблицу цветов. Например, если первому значению цвета в таблице цветов хранится значение RGB (255, 0, 0), то значению первого пикселя в растровом массиве будет поставлен в соответствие красный цвет. Значения пикселей хранятся в порядке их расположения слева на право начиная с нижней строки изображения. Таким образом в 256 цветном BMP файле первый байт данных растрового массива представляет собой индекс для цвета пикселя, находящегося в нижнем левом углу изображения.
Второй байт представляет индекс для соседнего справа пикселя и т.д. Если число байт в каждой строке нечетно, то каждой строке добавляется дополнительный байт, чтобы выровнять данные растрового массива по 16 битным границам.

Не все файлы BMP имеют структуру подобную указанной выше. Например, файлы BMP с глубиной 16 и 24 бита на пиксель не имеют таблиц цветов. В этих файлах значения пикселей растрового массива непосредственно характеризуют значения цветов. Также могут различаться внутренние форматы хранения отдельных разделов файла. Например, информация растрового массива в 16 и 256 битных файлах может сжиматься с помощью алгоритма RLE, который заменяет последовательности идентичных пикселей изображения на лексемы, определяющие число пикселей в последовательности и их цвет.

PCX
Файлы разделены на 3 части:

1. Заголовок
   Имеет размер 128 байт и содержит поля размера изображения и числа бит для кодирования информации для цвета каждого пикселя.
2. Данные растрового массива
   Сжимаются с помощью алгоритма RLE.
3. Факультативная таблица цветов
   Содержит 256 значений цветов RGB определяющих цвета изображения. Кодирование цвета каждого пикселя может производиться с глубиной 1, 4, 8 или 24 бита.

TIFF (Tagged Image File Format) Формат файлов изображения снабженных тегами
Файл начинается 8 байтовым заголовком (IFH), важнейший элемент которого – каталог файла изображения (Image File Directory) – служит указателем к структуре данных. IFD представляет собой таблицу для идентификации одной или нескольких порций данных переменной длины называемых тегами. Теги хранят информации об изображении. В спецификации формата файлов TIFF определено более 70 различных типов тегов. Например тег одного типа хранит информацию о ширине изображения в пикселях, другого о его высоте, в теге третьего типа хранится таблица цветов (при необходимости), а тег четвертого типа содержит сами данные растрового массива. Изображение закодированное в файле TIFF полностью определяется его тегами, и этот формат файла легко расширяется поскольку для придания файлу дополнительных свойств достаточно лишь определить дополнительные типы тегов.

В большинстве программ для чтения файлов TIFF реализуется только подмножество тегов именно поэтому вариант созданный одной программой файл TIFF иногда не может быть прочитан другой. Кроме того, программы создающие файлы TIFF могут определять собственные типы теги имеющие только для них. Программы для чтения TIFF файлов могут пропускать непонятные для них теги, но всегда существует опасность потерять внешний вид изображения.

Еще одна сложность заключается в том, что файл TIFF может содержать несколько изображений каждому из которых сопутствует собственный IFD и набор тегов. Данные растрового массива в файле TIFF могут сжиматься разными методами поэтому в надежной программе для чтения файлов TIFF должны быть средства распаковки из форматов RLE и LZW (Lempel Ziv Welch). Пользование программами распаковки LZW должно осуществляться в соответствии с

лицензионным соглашением LZW и часто за плату. В результате самые лучшие программы чтения TIFF файлов могут не справляться когда сталкиваются со сжатым по методам LZW изображением.

Несмотря на свою сложность формат TIFF остается одним из лучших типов для передачи растровых массивов с одной платформы на другую. Благодаря своей универсальности позволяющей кодировать в двоичном виде практически любое изображение без потери его визуальных и других атрибутов.

Графические протоколы — 2

12 декабря 2015 г.
10:05

GIF

GIF — Graphic Interchange Format
Существует две версии GIF-файлов: GIF 87a и GIF 89a.

GIF начинается с 13-байтового заголовка, содержащего сигнатуру, которая идентифицирует этот файл, в качестве GIF-файла, номер версии GIF, и другую информацию. Если файл хранит всего одно изображение, вслед за заголовком располагается общая таблица цветов. Если в файле хранится несколько изображений, то вместо общей таблицы цветов каждое изображение сопровождается локальной таблицей цветов

В файле GIF 87a вслед за заголовком и общей таблицей цветов размещается изображение, которое может быть первым из нескольких располагаемых подряд изображений. Каждое изображение состоит из локальной таблицы цветов и битов растрового массива, которые сжимаются с помощью LZW.

Файлы GIF 89a имеют аналогичную структуру, но они могут содержать дополнительные блоки с информацией о каждом изображении:

- Блоки-расширения для управления графикой, которые описывают как изображение должно выводиться на экран, накладываться предметы на предыдущее изображение;
- Блоки-расширения с обычным текстом;
- Блоки-расширения для комментариев;
- Блоки-расширения прикладных программ, в которых хранится информация, принадлежащая только создавшей этот файл программе.

Блоки-расширения могут находиться в любом месте файла после таблицы цветов.

Достоинства GIF:

- Широкое распространение формата и его компактность.

Недостатки GIF:

- Изображение не может быть больше 256 цветов;
- Проблемы лицензионного соглашения. Разработчики программ должны вносить плату за каждый экземпляр программы, использующей формат GIF.

PNG

PNG — Portable Network Graphic

Был разработан для замены GIF, чтобы обойти юридические препятствия, стоящие на пути использования GIF-файлов. PNG унаследовал многие возможности GIF и позволяет хранить информацию с истинными цветами.

JPEG

JPEG — Join Photographic Expert Group

Был разработал C-Cube Microsystems, как эффективный метод хранения изображений с большой глубиной цвета, например, получаемых при сканировании изображений или фотографировании с многочисленными, едва уловимыми оттенками цвета.

В отличие от других форматов в JPEG используется алгоритм сжатия с потерями информации. Алгоритм сжатия без потерь так сохраняет информацию об изображении, что распакованное изображение в точности соответствует оригиналу.

При сжатии с потерями часть информации утрачивается чтобы достичь большего коэффициента сжатия. Распакованное изображение JPEG редко соответствует оригиналу абсолютно точно, но часто эти различия настолько незначительны, что их едва можно обнаружить.

Процесс сжатия JPEG-изображения достаточно сложен и может потребовать большие вычислительные ресурсы для достижения приемлемой производительности.

Реальные изображения, в том числе фотографии, часто совсем невозможно сжать без потерь, поэтому сжатие на 50% считается достаточно хорошим. Существуют и другие методы сжатия без потерь, которые сжимают изображение на 90% (для специальных изображений), такие изображения плохо подходят для сжатия методом JPEG.

NURBS

12 декабря 2015 г.
10:57

NURBS — поверхности
Non-Uniform Rational B-splines

В современной компьютерной графике помимо полигональной технологии представления трёхмерных моделей используются и другие методы. В природе отсутствуют идеально прямые линии и плоскости, она, как правило, содержит сложные формы и кривые, поэтому для представления реальных объектов потребовался метод отличный от полигонального. Этот метод разрабатывался с 50-х годов инженерами, преследовавшими цель создать математическое описание поверхностей произвольной формы.

Созданием первого алгоритма был Пьер Безье из компании Рено. Его именем были названы т. н. кривые Безье (B-splines). Это математически рассчитанные кривые, плавно соединяющие отдельные точки. Существует фундаментальная теорема о том, что любую полиномиально заданную кривую можно разбить на кривые Безье. А сами кривые при этом легко поддаются алгоритмизации, поэтому идеально подходят для представления сложных прямых и поверхностей.

NURBS — это метод представления кривых через узловые точки. Некоторые из узлов можно наделять большим весом, тогда при изменении их координат, за ними, как за магнитом, тянутся другие точки. Если используется набор точек в пространстве, то можно получать криволинейную

поверхность. NURBS используется в автоматизации проектирования, в CAD-системах и позволяют инженерам создавать и визуализировать форму будущих изделий.

Также генерация поверхностей с помощью NURBS доступна в пакетах трёхмерного моделирования. Если поставить на поверхность несколько опорных точек и изменить положение одной из них, все остальные точки поверхности будут рассчитаны так, чтобы в поверхности не образовалось разрывов. Для NURBS пока не существует аппаратного ускорения, и поэтому они редко используются в приложениях реального времени.

РГЗ

21 ноября 2015 г.
11:21

~~На сцене изобразить стол (с наложенной текстурой)~~, ~~на столе разместить какие-нибудь объекты по своему усмотрению, но там обязательно должны быть несколько книг (с наложенными текстурами; разбросать или стопкой), шахматная доска и на ней несколько шахматных фигур (их можно нарисовать 3DMAX или найти готовые, но, чтобы каждая фигура была отдельная, т. е., чтобы доска с фигурой не были одной моделью)~~. ~~Сделать чтобы сцену можно было поворачивать, крутить и всё такое. Источники света: точечный (лучше несколько; чтобы можно было включать и выключать их клавишами; если найдём модель ламп, пусть это будет лампой), направленный, фоновый.~~

На автомат: ~~или сделать на сцене несколько полупрозрачных объектов (кубики или что-нибудь такое), расположить их как-нибудь на столе корректно (чтобы соблюдалась дальность-близость; это называется «сортировка по глубине»);~~ или сделать выделение объектов мышкой и переместить клавишами (т. е. щёлкаем по фигуре, она подсвечивается и мы её клавишами перетаскиваем) (это нахождение параметров селектирующего луча).