Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра ЭМИС

Буймов Б.А.

КОМПЬЮТЕРНАЯ ГРАФИКА

Лабораторные работы

Учебно-методическое пособие
для студентов специальности 080105 - Финансы и кредит и
для направлений: 080100 - Экономика, 080500 - Менеджмент

2011

Буймов Б.А.
Компьютерная графика. Лабораторные работы. Учебно-методическое пособие. - Томск: ТУСУР, 2011. - 24 с.

Учебно-методическое пособие посвящено рассмотрению математического аппарата, лежащего в основе компьютерной графики. В нем изложены способы представления на компьютере элементов графики, геометрических фигур, кривых различных классов, описаны математические методы преобразования координат, получения аксонометрических и перспективных проекций.

Пособие рассчитано на студентов вузов.

Подготовка и издание пособия выполнены за счет средств Программы «Инновационный Вуз».

Содержание

Введение

Этот курс лабораторных работ предназначен для практического изучения основ компьютерной графики. В лабораторных работах рассматриваются основные моменты курса лекций.

Описания лабораторных работ не содержат всего необходимого теоретического материала, а лишь основные моменты. Все работы выполняются на любом языке программирования процедурного типа (Си, Паскаль, ...). Рекомендации и примеры программ даны для Си с учетом Си++.

Лабораторная работа №1

Матричные преобразования

Для успешной дальнейшей работы вы должны построить свою небольшую графическую библиотеку, более высокого уровня, чем библиотека Си, а также построить свою матричную библиотеку.

Для открытия графического режима, для обработки ошибок приходится каждый раз писать одни и те же куски кода с небольшими вариациями. Поэтому удобно один раз написать одну универсальную функцию открытия графического режима и в дальнейшем пользоваться ей. Полезно также иметь функцию обработки графических ошибок.

Во многих случаях неудобно иметь дело со стандартной системой координат, когда начало координат находится в левом верхнем углу экрана. Поэтому неплохо иметь функцию установки своей системы координат и функции преобразования из одной системы в другую.

В данном курсе машинной графики во всех преобразованиях используются операции с матрицами. Для работы с ними необходимо иметь удобный набор функций.

Матрицы

Работа с матрицами характерна тем, что во время написания программы зачастую неизвестно количество матриц и их размеры. Это влечет за собой необходимость динамического распределения памяти. Для динамического управления памятью в Си существует стандартная библиотека функций.

Функции библиотеки динамического распределения памяти для Си:

```
void *calloc (size_t nitems, size_t size);
    // Выделяет память под nitems элементов
    // размером size.
void far *farcalloc(unsigned long nunits, unsigned long
unitsz);
    // Выделяет память под nitems элементов
    // размером size.
    // Возвращает дальний указатель.
    // Память выделяется вне сегмента данных и
    // размер блока может превышать 64K.
void *malloc(size_t size);
    // Выделяет память под nitems байт.
void far *farmalloc(unsigned long nbytes);
    // Выделяет память под nitems элементов.
    // Возвращает дальний указатель.
    // Память выделяется вне сегмента данных и
    // размер блока может превышать 64K.
void free(void *block);
    // Освобождает выделенную ранее память с помощью
    // функций calloc, malloc, realloc.
void farfree(void far *block);
    // Освобождает выделенную ранее память с помощью
    // функций farcalloc, farmalloc, farrealloc.
```

Все эти функции описаны в хедере alloc.h. В том случае, если вы работаете в Си++, то легче пользоваться операторами new и delete.

Функция создания матрицы должна, исходя из требуемого размера матрицы, выделять необходимое количество памяти и заполнять

соответствующую структуру данных. Структура данных, описывающая матрицу, может выглядеть например так:

```
struct  TMatrix
{
int  Rows;
int  Cols;
double  *Array;
};
```

Числа в матрице обязательно должны быть с плавающей точкой. Это необходимо для обеспечения достаточной точности преобразований, а так же и потому, что тригонометрические функции, используемые в преобразованиях, лежат в диапазоне от -1 до 1.

Наиболее приемлемый способ представления матрицы в памяти – размещение всей матрицы в непрерывной области памяти, при этом строки матрицы следуют друг за другом. Адрес элемента матрицы со строкой i и столбцом j вычисляется как:

```
Ptr + (i * SizeX) + j;
```

где Ptr - адрес первого элемента матрицы, SizeX - число столбцов в матрице.

Нежелательно хранить матрицы в виде массива строк, поскольку это приводит к значительной фрагментации памяти, и в конце концов может не найтись непрерывного куска памяти нужной длины. Кроме того, при таком способе представления легче сделать ошибки при работе с памятью.

Функции изменения матрицы могут быть очень разными, например:

- дополнение (исключение) строк (столбцов) в матрицу;
- склеивание матриц;
- разделение матриц;
- изменение размеров с копированием или без.

Функции изменения матриц не столь необходимы, поэтому реализовывать их необязательно, но функции склеивания и изменения размеров

матриц могут использоваться достаточно часто. Вы можете реализовать эти функции, если они вам понадобятся.

Для того, чтобы изменять отдельные элементы матриц в программе не заботясь при этом о способе представления матрицы, необходимо иметь функцию доступа к элементам матрицы. Эта функция должна получать матрицу, номер строки и столбца элемента. Возвращаемое значение - адрес нужного элемента (либо, если вы пользуетесь Си++, ссылка на нужный элемент). Можно поступить и по другому: создать две функции, одна из которых возвращает значение элемента, а другая записывает элемент в матрицу. Для удобства можно реализовать функции доступа таким образом, чтобы индексация элементов начиналась не с нуля, как принято в программировании, а с единицы, как принято в математике. Для этого перед обращением к памяти нужно номер строки и номер столбца уменьшить на единицу.

После того, как матрицы стали не нужны (например, перед выходом из программы) необходимо их уничтожить, т.е. освободить занимаемую память.

Естественно, что для этого нужно написать функцию уничтожения. В Си++ это будет деструктор.

Единственная матричная операция, которую мы будем использовать – это умножение. Результатом умножения матрицы $A(m \times n)$ на матрицу $B(n \times k)$ является матрица $C(m \times k)$. Обратите внимание на размерность результирующей матрицы. Правило умножения выглядит следующим образом:

$$C_{ij} = \sum_{l=1}^{n} A_{il} B_{ij}$$

Функция умножения должна быть насколько возможно быстрой, в этой функции не должно быть никаких лишних вычислений. Например, приведенная ниже реализация умножения никуда не годится:

```
/*.....  */
*(res.array + row*res.NofCols + col) = 0.0;
for(i=0; i<lMatr.NofCols; i++)
```

```
{
*(res.array + row*res.NofCols + col) +=
*(lMatr.array + row*lMatr.NofCols + i) *
*(rMatr.array + i*rMatr.NofCols + col);
}
```

Здесь кроме основной операции умножения производятся еще две операции умножения с индексами, что уменьшает быстродействие примерно в 1.5 – 2 раза. Чтобы избежать этого, лучше всего напрямую модифицировать указатели.

Поскольку для получения одного элемента результирующей матрицы нужно перемножить элементы строки левой матрицы на элементы столбца правой матрицы, то нужно получить адреса первых элементов строки и столбца и последовательно увеличивать их. Адрес строки увеличивается на 1, а адрес столбца - на число столбцов в правой матрице.

При преобразованиях объектов часто возникает необходимость выполнения операции типа $A = AT$, где A- матрица точек, а T- матрица преобразования. С точки зрения программирования эта операция выполняется следующим образом: конструируется временная матрица B, производится операция $B = AT$, B копируется в A, матрица B уничтожается. Чтобы реализовать такую последовательность действий необходимо иметь функцию копирования матриц. Эта функция должна копировать все элементы из одной матрицы в другую.

Кроме всех этих, совершенно необходимых функций, вам понадобятся функции считывания матриц из файла, а также функции вывода на экран и в файл. В файле матрицы удобно хранить в текстовом виде. Поскольку в одной программе может потребоваться несколько матриц, то целесообразно хранить их в одном файле. При этом различать матрицы можно по именам (они тоже хранятся в файле, непосредственно перед самой матрицей) или по номерам.

Предпочтительнее первый способ. Кроме того, в файле должны храниться размеры матрицы.

Задание

1. Создать структуру данных (для C++ класс), описывающую матрицу, и функции работы с матрицами. Минимальный набор функций:

- создание матрицы;
- уничтожение матрицы;
- чтение матрицы из файла;
- копирования матрицы;
- перемножение матриц;
- доступ к элементам.

Во всех функциях обязательно проверяйте соответствие размеров матриц и корректность индексов.

2. Написать функцию открытия графического режима, которая автоматически устанавливает драйвер и режим. В случае ошибки эта функция должна выводить сообщение об ошибке и прерывать выполнение программы.

3. Создать набор функций для преобразования системы координат. Необходимо обеспечить преобразование из декартовой системы координат в систему координат дисплея и обратно. При преобразовании учитывайте смещение центра координат, направление осей и масштаб по каждой оси. В этот набор также должна входить функция отображения координатных осей.

4. Для отладки и демонстрации этих функций напишите программу, которая считывает из файла две матрицы, перемножает их и выводит на экран. Для демонстрации графики и функций преобразования координат выведите на экран отрезок от точки (0, 0) до точки (1, 1). Центр координат должен быть в центре экрана, а масштабы заданы таким образом, чтобы ширина и высота экрана в декартовой системе координат равнялись 2.

Контрольные вопросы

1. Почему для копирования матрицы нужно писать отдельную процедуру? Нельзя ли просто скопировать одну структуру в другую?

2. Что может получиться, если в матричных операциях не проверять соответствие размеров и корректность индексов?

3. Каким образом будет вычисляться индекс нужного элемента в массиве, если матрица представлена в виде набора столбцов?

Лабораторная работа №2

Матричные преобразования на плоскости

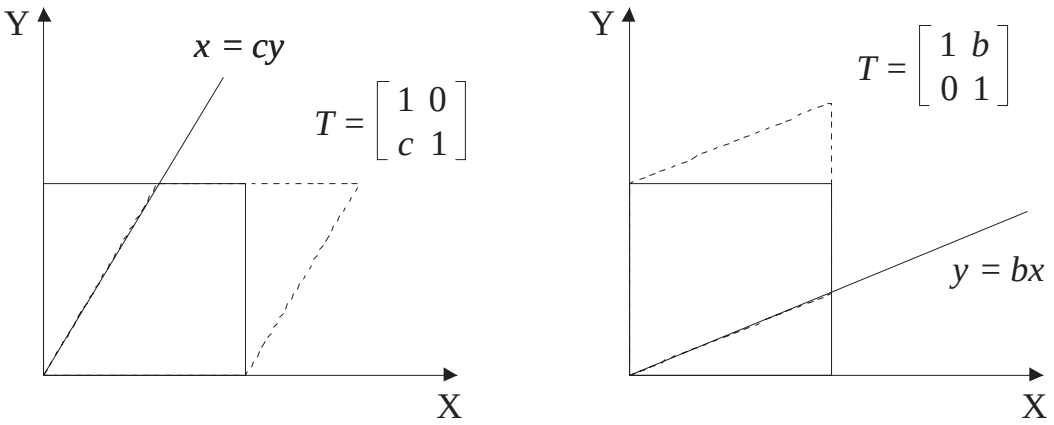
В этой работе вы изучите на практике матричные преобразования плоских векторных изображений.

В общем случае преобразование точки на плоскости выглядит следующим образом:

$$\begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} (ax + cy) & (bx + dy) \end{bmatrix}$$

Абсолютные значения коэффициентов a и d отвечают за изменение масштабов по координатным осям (a - по оси Ox, d - по оси Oy).

$$\begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} ax & dy \end{bmatrix};$$



a) горизонтальный сдвиг b) вертикальный сдвиг

Рисунок 1 – Преобразование сдвига

Знаки этих коэффициентов отвечают за преобразование симметрии относительно соответствующих осей. Если $a = -1$, то результирующая точка симметрична исходной относительно оси Oy. Если $d = -1$, то результирующая точка симметрична исходной относительно оси Ox.

Коэффициенты *b* и *c* отвечают за преобразование сдвига. Это преобразование иллюстрируется на рис. 1.

И, наконец, вращение относительно начала координат на угол Θ осуществляется с помощью матрицы вида:

$$T = \begin{bmatrix} \cos\Theta & \sin\Theta \\ -\sin\Theta & \cos\Theta \end{bmatrix}$$

Чтобы повернуть объект относительно любой другой точки, нужно перенести его в начало координат, повернуть и затем перенести обратно.

Недостатком преобразования на плоскости с помощью матрицы 2×2 является невозможность перемещения объекта без его искажения. Этот недостаток устраняется введением так называемых однородных координат. Но это тема следующей лабораторной работы.

Описание объекта

Для того, чтобы работать с графическими объектами, необходимо иметь описание этого объекта. Объект полностью описывается набором вершин и набором ребер. Таким образом, можно представить объект совокупностью двух матриц - матрицы вершин и матрицы ребер. Например, так:

```
struct TObject
{
TMatrix Vertexs; // матрица вершин
TMatrix Edges;  // матрица ребер
};
```

Все матричные преобразования применяются к матрице вершин. Матрица ребер используется при отображении объекта. В этой матрице хранятся номера концевых точек каждого ребра (одна строка соответствует одному ребру), и, следовательно, она имеет размер $n \times 2$, где n - число ребер.

Можно каждому ребру присвоить свой цвет, тогда получим матрицу $n \times 3$, где последний столбец содержит цвета ребер. В качестве матрицы ребер можно использовать обычную матрицу, но нужно следить, чтобы туда записывались только целые числа.

Объекты можно получать двумя способами - считывать из файла и конструировать непосредственно в программе. В первом случае из файла считываются две матрицы - матрица вершин и матрица ребер. Конструировать можно такие объекты, как правильные многоугольники.

Задание

Ваша задача - определить структуру данных, определяющую объект и написать следующие функции:

1) создания объекта;

2) уничтожения объекта;

3) чтения объекта из файла;

4) матричного преобразования объекта. В результате преобразования матрица точек заменяется на преобразованную матрицу.

5) отображения объекта. При отображении объекта необходимо каждую точку перевести из декартовой системы координат в систему координат дисплея, используя имеющуюся в вашем распоряжении функцию перевода координат.

С помощью написанных вами функций нарисуйте домик и примените к нему все описанные выше типы преобразований. При этом отображайте как преобразованный объект, так и исходный, причем преобразование должно вводиться плавно. Например, если это масштабирование, то размеры объекта должны увеличиваться (уменьшаться) постепенно.

Контрольные вопросы

1. Насколько невыгодно использовать обычные матрицы для хранения информации о ребрах? Можете ли вы предложить какой либо другой способ (кроме матриц)?

2. Что будет с единичным квадратом, лежащим в третьей четверти, если применить к нему преобразование сдвига?

3. Можете ли вы указать, с помощью какого преобразования получен шрифт, которым набраны названия лабораторных работ?

4. Каким образом можно переносить объекты?

Лабораторная работа №3

Преобразования в однородных координатах

В однородных координатах преобразование определяется матрицей

$$T = \begin{bmatrix} a & b & p \\ c & d & q \\ m & n & s \end{bmatrix}$$

Коэффициенты a и d имеют тот же смысл, что и в неоднородном преобразовании. В однородных координатах мы получаем возможность переносить объект без искажений и, как следствие вращать его вокруг любой точки. Коэффициенты m и n отвечают за перемещение соответственно по координатам x и y. С помощью коэффициента s можно производить общее масштабирование объекта.

Для вращения объекта вокруг произвольной точки нужно переместить его так, чтобы точка вращения оказалась в начале координат, затем повернуть его на нужный угол и переместить в исходную точку. Полная матрица вращения объекта на угол Θ относительно точки $[m \times n]$ получается как произведение трех матриц:

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -m & -n & 1 \end{bmatrix} \times \begin{bmatrix} \cos\Theta & \sin\Theta & 0 \\ -\sin\Theta & -\cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{bmatrix}$$

Перед отображением преобразованного объекта необходимо нормализовать координаты объекта, т.е. каждую строку матрицы вершин разделить на последнюю компоненту. Для этой цели напишите функцию нормализации матрицы. Не забывайте, что теперь вектор-точка имеет три компоненты.

Функция нормализации должна работать с матрицами, имеющими любое количество столбцов - это нужно для последующих работ с трехмерными объектами.

Задание

Напишите программу, которая вращает домик из предыдущей работы вокруг начала координат. При этом домик должен сохранять вертикальное положение, как на рис. 2. Для этого перед поворотом (или после поворота) вокруг начала координат на угол φ. домик нужно повернуть вокруг его центра (точка A) на угол $-\varphi$. Добейтесь того, чтобы размеры домика изменялись пропорционально величине $c + \sin\varphi$, где c - некоторая константа, например 1. Чтобы радиус вращения не изменялся при изменении размеров, перед масштабированием нужно перенести домик так, чтобы конец радиуса - точка A- совместился с началом координат. После масштабирования нужно выполнить обратный перенос.

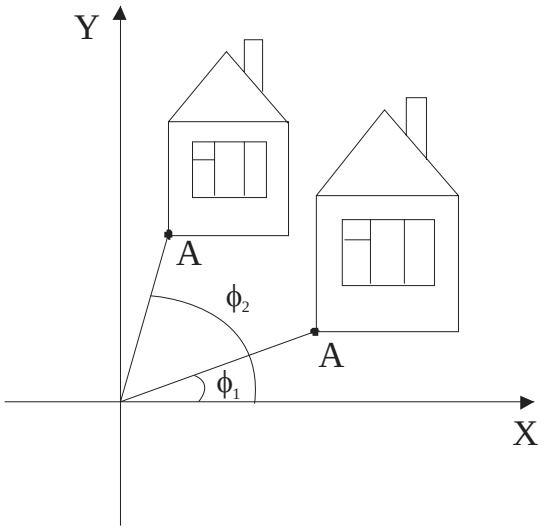


Рисунок 2

Контрольные вопросы

1. Чем отличается перенос объектов в однородных координатах от той же операции в неоднородных координатах?

Лабораторная работа №4

Трехмерные преобразования

В этой работе мы рассмотрим трехмерные преобразования в однородных координатах.

Матрица преобразования имеет вид:

$$T = \begin{bmatrix} a & b & c & p \\ d & e & f & q \\ h & i & j & r \\ l & m & n & s \end{bmatrix}$$

Матрица 3×3 в левом верхнем углу осуществляет преобразования покоординатного масштабирования, сдвига и вращения. Элементы l, m и n отвечают за перенос, а элементы p, q и r - за перспективные преобразования.

Матрица вращения вокруг произвольной оси получается следующим образом:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -l & -m & -n & 1 \end{bmatrix} \times R \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ l & m & n & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} n_1^2 + (1 - n_1^2)\cos\alpha & n_1 n_2 (1 - \cos\alpha) + n_3 \sin\alpha & n_1 n_3 (1 - \cos\alpha) - n_2 \sin\alpha & 0 \\ n_1 n_2 (1 - \cos\alpha) - n_3 \sin\alpha & n_2^2 + (1 - n_2^2)\cos\alpha & n_2 n_3 (1 - \cos\alpha) + n_1 \sin\alpha & 0 \\ n_1 n_3 (1 - \cos\alpha) + n_2 \sin\alpha & n_2 n_3 (1 - \cos\alpha) - n_1 \sin\alpha & n_3^2 + (1 - n_3^2)\cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

где n_1, n_2 и n_3 - направляющие косинусы оси вращения;

α - угол поворота;

l, m и n - координаты какой-либо точки на оси вращения.

Задание

Напишите функцию, которая конструирует матрицу вращения. В эту функцию будут передаваться центр вращения и вектор направления оси вращения.

Вектор направления должен иметь единичную длину. Длина l вектора $[x\ y\ z]$ вычисляется, как $l=\sqrt{x^2 + y^2 + z^2}$. Чтобы не думать каждый раз о длине вектора, напишите функцию нормализации вектора, для этого каждую компоненту вектора нужно разделить на его длину.

Создайте объемный домик, и с помощью новой функции повращайте его. Чтобы менять центр вращения и вектор направления оси вращения, не изменяя текста программы, запишите их в файл.

Для отображения трехмерного объекта на плоскость используйте любые две координаты из трех (например x и y), таким образом вы получите простейшую прямоугольную проекцию.

Контрольные вопросы

1. Каким образом повлияет на вращение изменение направления оси вращения на противоположное?

2. Можно ли вращать саму ось вращения?

Лабораторная работа №5

Проекции

Проекции - совершенно необходимый элемент трехмерной графики. Они позволяют наблюдать трехмерные объекты на плоскости (т.е. на экране). С помощью отдельных проекций для каждого глаза можно создавать иллюзию объемности.

Основные виды проекций - аксонометрические и перспективные.

Аксонометрические проекции в свою очередь делятся на:

1) ортогональные - проекции на координатные плоскости;

2) диметрические - система координат поворачивается так, что две оси из трех одинаково сокращаются. Получается последовательными поворотами вокруг двух осей. Если сначала повернуть систему координат вокруг оси Oy на угол φ а затем вокруг оси Ox на угол α, то

$$\sin^2\varphi = \frac{\sin^2\alpha}{1-\sin^2\alpha} = \frac{\sin^2\alpha}{\cos^2\alpha} = tg^2\alpha$$

Обычно углы φ. и α. выбирают таким образом, чтобы третья ось (Oz) сокращалась в определенной степени. Например, если ось Oz сокращается в два раза, то $\sin^2\alpha = 1/8$ а $\sin^2\varphi = 1/7$;

3) изометрическая - все три координатные оси одинаково сокращены – частный случай диметрической проекции. Здесь $\varphi = 45°$, а $\sin\alpha = 1/\sqrt{3}$.

Перспективные преобразования получаются при ненулевых коэффициентах p, q и r. Значения этих коэффициентов - это точки схода на соответствующих осях. Например, матрица

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

осуществляет перспективное преобразование с центром в точке $[0\ 0\ -1/r]$ и точкой схода $[0\ 0\ r]$.

Применяя перспективные преобразования, нужно следить за тем, чтобы объект всегда находился по одну сторону от центра проекции. При построении неподвижных сцен полезно сочетать перспективу с предварительными поворотами и смещениями - другими словами, выбрать наилучшую точку наблюдения. В динамических сценах расположение наблюдателя и направление взгляда могут меняться (например, камера следит за объектом).

Чтобы получить эффект «точки зрения», нужно переместить всю сцену таким образом, чтобы точка зрения совпала с началом координат.

Направление взгляда задается вектором. После выбора точки зрения нужно повернуть сцену так, чтобы направление вектора взгляда совпадало с отрицательным направлением оси Oz. Если направление взгляда задается вектором $[\alpha\ \beta\ \gamma]$ единичной длины, нужно повернуть сцену относительно вектора $[\beta\ -\alpha\ 0]$ на угол $\varphi = \arccos\gamma$.

Задание

1. Постройте все четыре вида проекций вашего домика и выведите их на экран (все сразу).

2. Поставьте ваш домик на сетчатое поле и посмотрите на него с различных точек, используя перспективное преобразование. Еще лучше, если вы предусмотрите в вашей программе возможность изменения точки наблюдения, направления взгляда и параметры перспективы расстояния.

Контрольные вопросы

1. В каких случаях применяются аксонометрические проекции, а в каких перспективные?

2. Какой будет визуальный эффект, если в матрице перспективной проекции коэффициент r поменять на $-r\ (p = q = 0)$?

Лабораторная работа №6

B-сплайны

B-сплайны наиболее подходящие для изучения кривые, они сочетают в себе свойства многих видов кривых. Поэтому на примере этого типа кривых мы рассмотрим все другие типы.

B-сплайны, также как и кривые Безье, определяются характеристическим многоугольником. Но в отличие от кривых Безье порядок кривой можно менять. Это достигается введением кратных вершин характеристического многоугольника и вектора параметрических узлов. Итак, B-сплайн описывается следующим набором параметров:

n - число вершин характеристического многоугольника без единицы;

k - порядок кривой;

P - вершины характеристического многоугольника;

$N_{i,k}$ - весовые функции, где i - номер вершины;

x - вектор параметрических узлов.

Сама кривая задается соотношением

$$P(t)=\sum_{i=0}^{n} P_i N_{i,k}(t)$$

Весовые функции определяются рекуррентными формулами

$$N_{i,k}(t)=\begin{cases}1, & t\in[x_i, x_{i+1}]\\ 0, & t\notin[x_i, x_{i+1}]\end{cases} \text{ и}$$

$$N_{i,k}(t)=\frac{(t-x_i)N_{i,k-1}(t)}{x_{i+k-1}-x_i}+\frac{(x_{i+k}-t)N_{i+1,k-1}(t)}{x_{i+k}-x_{i+1}}$$

Вектор параметрических узлов x определяется числом вершин характеристического многоугольника, наличием кратных вершин и порядком кривой. Он содержит $n+k+1$ элементов и вычисляется по следующему алгоритму:

1) для $0\le i\le k-1 \quad x_i=0$;

2) для $\mathrm{k}\le \mathrm{i}\le \mathrm{n}+1$, если $P_{i-k}\neq P_{i-k+1}$, то $x_i=x_{i-1}+1$ иначе $x_i=x_{i-1}$;

3) для $n+2 \le i \le n+k \quad x_i = x_{i-1}$

Для вычисления весовых функций $N_{i,k}$, используйте вспомогательную матрицу размером $n+2 \times k$. Например для сплайна с 4-мя характеристическими вершинами 3-го порядка матрица весовых функций выглядит так:

$$\begin{bmatrix} {}^{1}N_{0,1} & {}^{6}N_{0,2} & {}^{11}N_{0,3} \\ {}^{2}N_{1,1} & {}^{7}N_{1,2} & {}^{12}N_{1,3} \\ {}^{3}N_{2,1} & {}^{8}N_{2,2} & {}^{13}N_{2,3} \\ {}^{4}N_{3,1} & {}^{9}N_{3,2} & {}^{14}N_{3,3} \\ {}^{5}N_{4,1} & {}^{10}N_{4,2} & \end{bmatrix}$$

В левом верхнем углу каждой позиции указан порядковый номер весовых функций при вычислении. Последняя строка этой матрицы служит только для вычисления предыдущей строки, и последний элемент вычислять не нужно.

Искомый результат - это последний столбец матрицы.

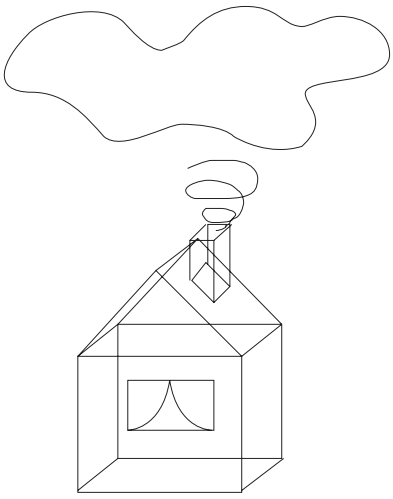


Рисунок 3

Для работы с B-сплайнами вам понадобится соответствующая структура данных и функции работы с ними. Структура данных должна содержать все параметры сплайна, и возможно какие-либо дополнительные, например цвет.

Для описания матричных параметров естественно использовать имеющиеся в вашем распоряжении матрицы. Вершины характеристического многоугольника для наших целей наиболее удобно хранить в файле, в

текстовом виде - в этом случае вы получите возможность легко изменять параметры кривой.

Для пространственного преобразования достаточно матрицу характеристического многоугольника умножить на матрицу преобразования.

Задание

Напишите функцию отрисовки B-сплайна, которая будет строить саму кривую и характеристический многоугольник. Характеристический многоугольник должен отличаться от самой кривой; для этого его можно рисовать другим цветом и/или пунктиром. Поэкспериментируйте с различными параметрами сплайна.

С помощью сплайнов добавьте к вашему домику шторки на окна, дым из трубы и облака над домиком, как это показано на рис. 3.

Контрольные вопросы

1. В каком диапазоне может лежать порядок B-сплайна?

2. Как влияют порядок и кратность на форму кривой? Покажите это.

3. Как добиться того, чтобы в месте стыка двух сплайновых сегментов (или в точке соединения концов замкнутого сегмента) не было излома.

Алфавитный указатель

Рекомендуемая литература

1. Роджерс Д., Адамс Дж. Математические основы машинной графики. Пер. с англ./ Дэвид Ф. Роджерс, Дж. Алан Адамс; Пер. П.А.Монахов, Пер. Г.В.Олохтонова, Пер. Д.В.Волков. М.: Мир, 2001. - 605[3] с.:а-ил.
2. Роджерс Д. Алгоритмические основы машинной графики. М.: Мир, 1989 - 512 с.
3. Фоли Дж., Вэн Дем А. Основы интерактивной машинной графики. Кн. 1 и 2. М.: Мир, 1985.
4. Павлидис Т. Алгоритмы машинной графики и обработки изображений. М.: Радио и Связь, 1986 - 400 стр.
5. Ростков А.А. Создайте анимацию сами. М.: ДИАЛОГ-МИФИ, 1995.
6. Ласло М. Вычислительная геометрия и компьютернаяграфика на С++ /Перевод с англ. М.: БИНОМ, 1997. - 304 с.: ил.
7. Корриган Дж. Компьютерная графика: Секреты и решения. М.: Энтроп, 1995. - 350 с.: ил.
8. Иванов В.П., Батраков А.С. Трехмерная компьютерная графика /Под ред. Г.М.Полищука. М.: Радио и связь, 1995. - 224 с.: ил.
9. Шикин Е.В., Боресков А.В. Компьютерная графика.Динамика, реалистические изображения. М.: "ДИАЛОГ-МИФИ", 1995. - 288 с.: ил.