1

О названии игры

Эта книга посвящена компьютерной системе METAFONT, которая тесно связана с издательской системой TEX, подробно описанной в книге *Все про TEX*. Каждая из этих систем отвечает за одну из двух основных задач печати: TEX правильно размещает символы на странице, а METAFONT определяет форму самих символов. Если их пути когда-нибудь и разойдутся, то это произойдет очень нескоро.

Почему же система называется 'METAFONT'? Часть '-FONT' легко объяснима, поскольку происходит от английского слова font — "шрифт", а шрифтами принято называть используемые в печати множества взаимосвязанных символов. Часть 'META-' более интересна. Она указывает на то, что нас интересует описание шрифтов на уровне, более высоком по сравнению с описанием любого конкретного шрифта.

Вообще, новообразованные слова, имеющие приставку "мета-", отражают современную тенденцию к рассмотрению вещей как бы извне или сверху, на более абстрактном уровне, соответствующем, как нам кажется, более зрелому пониманию. Так, в наши дни существуют метапсихология (наука о том, как связаны сознание и тело, в котором оно существует), метаистория (наука о принципах, определяющих общий ход событий), метаматематика (наука о математическом обосновании), металитература (литература, в которой описаны ее же формы) и т.п. Метаматематик занимается доказательством метатеорем (теорем о теоремах); ученые-информатики часто работают с метаязыками (языками, служащими для описания языков). Точно так же, металрифт представляет собой схематическое описание различных начертаний в пределах семейства родственных шрифтов; начертания шрифтов изменяются в соответствии с изменением определяющих параметров.

Метадизайн значительно сложнее простого дизайна, ведь куда проще нарисовать что-либо, чем объяснить, как это сделать. Одно из затруднений состоит в том, что предусмотреть все возможные описания непросто. Другое — в том, что компьютеру нужно объяснять абсолютно все. Но если нам один раз удалось описать достаточно общим образом, как рисовать нечто, то это же описание подойдет и при других обстоятельствах для аналогичных объектов. Поэтому оказывается, что время, затраченное на точное описание, себя оправдывает.

Шрифты, используемые для набора текста, как правило, достаточно мелки, и глазу легче воспринимать их, если буквы разработаны с учетом размера, в котором они будут реально использоваться. Получить шрифт размера 7 пунктов можно, просто уменьшив шрифт размера 10 пунктов до 70% натуральной величины. Но, хотя это и кажется на первый взгляд заманчивым, выбирая этот легкий путь, мы значительно проигрываем в качестве. Гораздо лучших результатов можно добиться, введя в правила метадизайна параметрическую изменчивость. Встроенная изменчивость имеет преимущества даже при разработке одного шрифта фиксированного размера, поскольку позволяет отложить принятие решений по многим вопросам на потом. Оставив на ранней стадии некоторые величины неопределенными, трактуя их как параметры, а не "замораживая", вы позволите компьютеру изобразить множество вариантов, соответствующих различным значениям параметров, и сможете затем сравнить результаты этих экспериментов. Такой подход значительно расширяет возможности изменения и настройки внешнего вида символов.

Но если металрифты имеют столь значительные преимущества перед старыми добрыми обычными шрифтами, то почему их не разработали раньше? Главная причина в том, что до недавнего времени компьютеров попросту не существовало. А для людей вычисления со множеством параметров были делом трудным и уто-

мительным. Сегодня же эту работу с легкостью проделывают машины, и введение параметров — это естественный продукт автоматизации.

Ну, хорошо, согласимся, что, по крайней мере теоретически, металрифты — вещь хорошая. Однако пока непонятно, как они должны быть устроены практически. Как задать зависимость начертания от неопределенных параметров?

Если переменным является лишь один параметр, то проблему достаточно легко решить визуально, накладывая друг на друга ряд изображений, в которых изменения начертания представлены графически. Например, если параметр меняется в пределах от 0 до 1, мы можем заготовить пять набросков, соответствующих значениям параметров 0, $\frac{1}{4}$, $\frac{1}{2}$, $\frac{3}{4}$ и 1. Располагая эти наброски по порядку, мы можем без труда выполнить интерполяцию и найти начертание, отвечающее значению, которое находится в промежутке между двумя заданными, например, $\frac{2}{3}$. Мы даже можем попробовать выполнить экстраполяцию, например, для значения $1\frac{1}{4}$.

Но если число независимых параметров два и более, то визуальное решение становится слишком громоздким. Мы вынуждены применить "вербальный" подход, используя для описания искомых изображений некий язык. Представим, например, что мы хотим объяснить другу, который находится в далекой стране, как выглядит буква 'a' некоторого шрифта, используя для общения только телефон. Желательно, чтобы наш друг в точности воспроизвел то начертание, которое мы ему описали. Если нам удастся сделать это для какого-то одного начертания, то не составит труда обобщить наше устное описание, введя вместо констант переменные параметры.

Поясним сказанное, прибегнув к аналогии с кулинарией. Представьте, что вы испекли превкусный пирог с ягодами, и друзья попросили его рецепт, чтобы самим испечь такой же. Если вы привыкли готовить, целиком полагаясь на интуицию, то записать в точности всю последовательность своих действий может оказаться непросто. Однако существует традиционный язык описания рецептов, и если вы прибегнете к скрупулезным измерениям, то окажется, например, что вы использовали $1\frac{1}{4}$ стакана сахара. Если бы вы подобным образом инструктировали кухонный комбайн, управляемый при помощи компьютера, то могли бы на следующем этапе перейти к метарецепту. В нем вы указали бы, например, что на каждые x стаканов ягод следует использовать $.25x$* стаканов сахара или $.3x + .25y$ стаканов сахара — на x стаканов черники и y стаканов ежевики.

Иными словами, переход от дизайна к метадизайну имеет много общего с переходом от арифметики к элементарной алгебре. Числа заменяются простыми формулами, содержащими неизвестные величины. В дальнейшем мы не раз столкнемся с примерами подобной замены.

В системе METAFONT определение полноценного шрифта состоит из трех основных частей. Первую составляют довольно рутинные процедуры, выполняющие необходимые технические функции, такие, например, как присваивание кодовых чисел конкретным символам и позиционирование каждого символа внутри невидимой рамки. Последнее необходимо для того, чтобы программа верстки смогла правильно расположить символы друг относительно друга в наборе. Далее следует более интересный набор процедур, которые предназначены для рисования основных элементов, характерных для символов данного шрифта (засечек, скруглений, петель, арок и т.п.). В описания этих процедур, как правило, входят их собственные

---

* Заметьте, что для отделения целой части десятичного числа от дробной используется не запятая, а *точка*, поскольку в системе METAFONT десятичные числа представляются именно так. — *Прим. перев.*

специальные параметры, так что каждая из них позволяет воспроизводить множество родственных элементов. Так, например, при помощи одной и той же процедуры можно рисовать засечки различной длины, хотя во всех них будет угадываться один почерк. И наконец, в определении шрифта содержатся программы для каждого из символов. Если процедуры первого и второго типа выбраны удачно, то программы для символов представляют собой достаточно простые описания высокого уровня, не обремененные излишними подробностями. Это позволяет получать шрифты различного вида, изменяя только процедуры (например, процедуру, рисующую засечки) и не меняя при этом ни одной из программ, в которых эти процедуры используются. [Особо впечатляющий пример использования такого подхода продемонстрировали Джон Д. Хобби и Гу Гуоянь в своей работе "A Chinese Meta-Font" (*TUGboat* **5** (1984), 119–136). Изменяя лишь 13 базовых подпроцедур и используя одни и те же программы для символов, им удалось воспроизвести 128 основных китайских иероглифов в трех различных начертаниях (Song, Long Song и Bold).]

При помощи хорошей программы для METAFONT дизайнер может выразить свой замысел намного точнее, чем при помощи рисунка, поскольку в языке алгебры присутствуют простые идиомы, которые позволяют выразить многие визуальные связи между объектами. Таким образом, программы для METAFONT можно использовать для передачи информации о том, как должны выглядеть символы, точно так же, как опыт искусного повара можно передать посредством рецептов. Однако голые алгебраические формулы воспринимаются с трудом. Описания METAFONT нуждаются в сопутствующих иллюстрациях так же, как построения из учебников по геометрии — в сопутствующих чертежах. Глупо ожидать, что, прочитав текст программы для METAFONT, кто-либо воскликнет: "Какая красивая буква!". Но если имеется одно или несколько увеличенных изображений этой буквы, то глядя на них, человек, прочитавший соответствующую программу, может с полным правом заявить: "Я знаю, как нарисована эта красивая буква!". В дальнейшем мы увидим, что система METAFONT позволяет легко получать "пробные оттиски", которые здорово помогают в работе над программой.

Несмотря на то что язык METAFONT предназначен для описания металрифтов, его, конечно, можно использовать и для описания фиксированных начертаний, для которых приставка "мета-" теряет смысл. Более того, METAFONT можно использовать не только для разработки шрифтов; система позволяет вполне успешно изображать геометрические объекты, никак не связанные с символами или иероглифами. Автор, например, иногда использует METAFONT как калькулятор, выполняя в интерактивном режиме простые вычисления. Компьютеру ведь безразлично, что программы используются для целей, которые не соответствуют их названиям.

Комбинируя постоянные и переменные виды начертаний [Тингуэли],
создал несколько крупных, красочных открытых рельефов.
Впоследствии, для того чтобы подчеркнуть идеи и стили,
положенные в основу их концепций, он назвал эти рельефы
Meta-Kandinsky и Meta-Herbin.

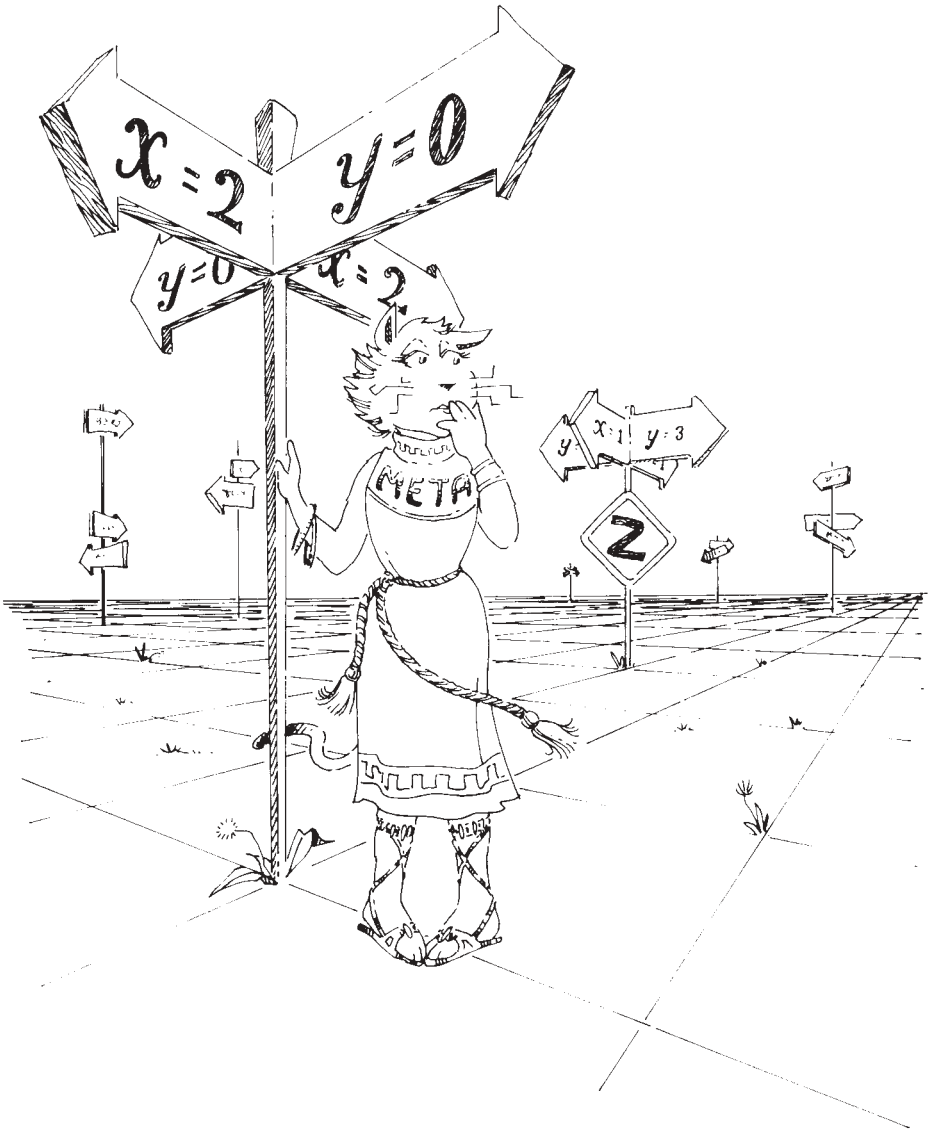
— К. Г. ПОНТУС ХАЛЬТЕН, *Джин Тингуэли: Мета* (1972)

Сама идея металрифта теперь понятна. Но в чем ее прелесть?
Возможность манипулирования множеством параметров, очевидно,
интересна и забавна, но кому нужен шрифт размера 6$\frac{1}{7}$ пункта,
представляющий собой нечто среднее между Baskerville и Helvetica?

— ДОНАЛЬД Э. КНУТ, *Понятие металрифта* (1982)

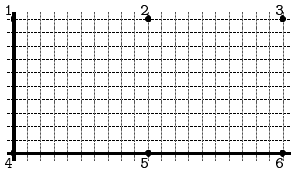
2

Координаты



Если мы хотим инструктировать компьютер, как он должен изображать ту или иную фигуру, необходимо объяснить ему, где должны быть расположены ее ключевые точки. Для этого в системе METAFONT используются обычные *декартовы координаты*. Положение точки определяется заданием координаты x — числа единичных отрезков вправо от точки отсчета (*reference point*), и координаты y — числа единичных отрезков вверх от точки отсчета. Вначале мы определяем горизонтальную (вправо/влево) составляющую положения точки, а затем — вертикальную (вверх/вниз). Мир METAFONT — двумерный, поэтому двух координат достаточно.

Рассмотрим, например, следующие шесть точек:



В системе METAFONT их положения задаются следующим образом:

$$(x_1, y_1) = (0, 100); \quad (x_2, y_2) = (100, 100); \quad (x_3, y_3) = (200, 100);$$
$$(x_4, y_4) = (0, \ \ 0); \quad (x_5, y_5) = (100, \ \ 0); \quad (x_6, y_6) = (200, \ \ 0).$$

Точка 4 совпадает с точкой отсчета, поскольку обе ее координаты — нули; для того чтобы попасть в точку $3 = (200, 100)$, нужно сделать 200 шагов вправо от точки отсчета и 100 шагов вверх, и так далее.*

▸ **Упражнение 2.1**
Какая из приведенных выше шести точек ближе всего к точке $(60, 30)$?

▸ **Упражнение 2.2**
Верно ли, что все точки, лежащие на одной горизонтальной прямой, имеют одинаковую координату x?

▸ **Упражнение 2.3**
Где расположена точка $(-5, 15)$?

▸ **Упражнение 2.4**
Каковы координаты точки, лежащей на 60 единичных отрезков ниже точки 6 на приведенной выше диаграмме? ("Ниже" означает не "под страницей", а "вниз по странице".)

При работе с METAFONT, прежде чем перейти к определению фигуры, вы обычно составляете ее примерный набросок на миллиметровой бумаге и нумеруете важные точки этого наброска любым удобным для вас образом. Затем вы пишете программу для METAFONT, в которой задаете (а) координаты этих ключевых точек и (б) прямые или кривые линии, которые должны соединять эти точки.

В METAFONT имеется своя встроенная "миллиметровка", которая образует так называемый растр (*raster*), или сетку, состоящую из квадратиков-пикселей. Вывод METAFONT определяет цвет пикселей. Некоторым из них будет приписан черный цвет, а остальным — белый. Таким образом, действия компьютера сводятся в основном к преобразованию рисунков в таблицы из нулей и единиц, на основе которых разработчик может восстановить рисунок, закрашивая соответствующие пиксели. Этот процесс напоминает вышивку по канве нитками двух цветов.

---

* Заметьте, что в системе METAFONT координаты точки отделяются друг от друга не точкой с запятой, а *запятой*. — *Прим. перев.*

Координаты определяют длину, но мы пока не говорили, в каких единицах они выражаются. Важно выбрать удобные единицы измерения. В системе METAFONT координаты задаются в пикселях. Таким образом, квадратики в приведенном выше рисунке, соответствующие разнице в 10 единиц по горизонтали и вертикали, представляют массивы пикселей размера 10×10, а прямоугольник, ограниченный шестью отмеченными точками, содержит в общей сложности 20 000 пикселей.*

Координаты не обязательно должны быть целыми числами. Можно, например, сослаться на точку $(31.5, 42.5)$, лежащую в самом центре пикселя с координатами вершин $(31, 42)$, $(31, 43)$, $(32, 42)$ и $(32, 43)$. Компьютер может обрабатывать координаты, значения которых представляют собой целое число, умноженное на $\frac{1}{65536} \approx 0.00002$ от ширины пикселя, что обеспечивает высокую точность построений. Однако METAFONT никогда не закрашивает часть пикселя, а только весь пиксель целиком; тут уж — все или ничего.

Размеры пикселей определяются *разрешением* (*resolution*) растровой сетки. Разрешение измеряется, как правило, в пикселях на дюйм (в Америке), и в пикселях на миллиметр (везде, кроме Америки). К примеру, шрифт, которым набран этот текст, разработан при помощи METAFONT с разрешением, чуть большим 700 пикселей на дюйм, но чуть меньшим 30 пикселей на миллиметр. Пока мы будем считать, что размеры пикселей столь ничтожны, что округление до целого числа пикселей ни на что серьезно не влияет. Позже мы обсудим проблемы, связанные с округлением, которые возникают при работе с малым разрешением.

Программы для METAFONT желательно составлять так, чтобы они позволяли генерировать шрифты для устройств с различной разрешающей способностью. Тогда пробные оттиски символов, разработанных для устройств с высоким разрешением, можно будет получать на печатающих устройствах с низким разрешением. Поэтому в программах для METAFONT ключевые точки редко определяются указанием точных числовых значений, например, '100'; как правило, координаты определяются по отношению к некоторой величине, зависящей от разрешения, что позволяет с легкостью вносить изменения. Например, координаты шести рассмотренных выше точек лучше определить следующим образом:

$$(x_1, y_1) = (0, b); \qquad (x_2, y_2) = (a, b); \qquad (x_3, y_3) = (2a, b);$$
$$(x_4, y_4) = (0, 0); \qquad (x_5, y_5) = (a, 0); \qquad (x_6, y_6) = (2a, 0).$$

Тогда величины a и b можно будет выбрать с учетом нужного разрешения. В предыдущем примере мы имели: $a = b = 100$. Но такие постоянные значения уменьшают гибкость программы либо не оставляют ее вовсе.

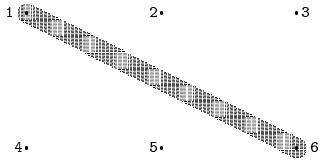
Обратите внимание на величину '$2a$' в определениях точек x_3 и x_6; METAFONT достаточно хорошо знает алгебру, для того чтобы сообразить, что это означает значение a, умноженное на два, каково бы ни было это значение. В главе 1 мы отмечали, что именно использование простых алгебраических операций и делает METAFONT метасистемой. Интересно, что декартовы координаты названы в честь Рене Декарта (René Descartes) не потому, что он их выдумал, а потому, что он показал, насколько эффективно можно их использовать, применяя алгебраические методы. Координаты использовались для таких целей, как определение широты и долготы задолго до Декарта, но именно он первым заметил, что, подставляя в коор-

---

* Как правило, термин "пиксель" обозначает квадратный фрагмент рисунка, но иногда мы используем его для обозначения единицы длины. Пиксель-квадратик имеет один линейный пиксель в длину и один линейный пиксель в высоту.

динаты переменные величины, мы получаем возможность описывать бесконечные множества взаимосвязанных точек и выводить свойства кривых, что было весьма затруднительно при использовании одних лишь геометрических методов.

Мы уже определили несколько точек, но пока ничего с ними не делали. Давайте нарисуем прямую, соединяющую точки 1 и 6, следующего вида:



Один из способов сделать это при помощи METAFONT — указать в программе

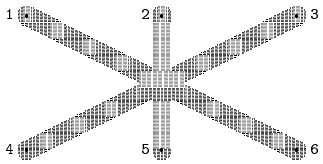
draw (x_1, y_1) .. (x_6, y_6).

Здесь посредством '..' мы просим компьютер соединить две точки.

Такие формулы, как '(x_1, y_1)', приходится писать довольно часто. Поэтому для них удобно ввести краткое обозначение. Впредь мы будем обозначать точку (x_1, y_1) через z_1, и вообще, через z_k с произвольным индексом — точку (x_k, y_k). Таким образом, приведенную выше команду '**draw**' можно переписать как

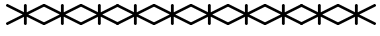
draw z_1 .. z_6.

Добавив посредством команд '**draw** z_2 .. z_5' и '**draw** z_3 .. z_4' еще две прямые, мы получаем фигуру, напоминающую узор на флаге Великобритании (Union Jack):



Мы назовем этот символ "гексом" (*hex*), так как он имеет шесть концов. Отметим, что линии имеют некоторую толщину и скруглены на концах так, будто нарисованы чувствительным к наклону пером с круглым кончиком. METAFONT обеспечивает множество средств для регулирования толщины линий и изменения формы их концов, однако эти возможности мы обсудим позже, поскольку сейчас нашей главной целью является изучение координат.

Если символ гекс уменьшить так, чтобы параметр b, определяющий его высоту, равнялся высоте букв в этом абзаце, он будет иметь вид '  '. Просто ради интереса попробуем напечатать подряд десять таких символов:



До чего же это просто!*

---

* Теперь, когда мы получили возможность с легкостью вводить новые символы и использовать их затем в своих печатных трудах, следует задуматься, как далеко мы желаем зайти в своих экспериментах? Разумно ли тысячами вводить новые символы? Не приведет ли это к появлению шрифтов-уродцев? Такие вопросы выходят за рамки этой книги. Однако легко вообразить, какая эпидемия шрифтомании может вспыхнуть, если люди поймут, насколько увлекательно конструировать собственные символы. Тогда, пожалуй, придется лечить их при помощи специально разработанной операции "шрифтолоботомии".

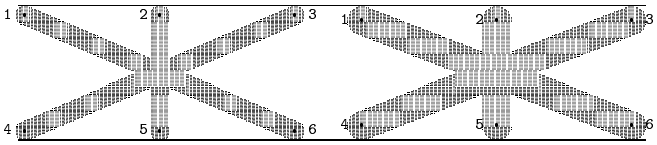
Давайте-ка приглядимся к нашему новому символу. Символ ⋊ слегка высоковат, поскольку выходит за пределы точек 1, 2 и 3, если учитывать толщину линий. Кроме того, он опускается чуть ниже базовой линии (*baseline*), т.е. прямой с координатой $y = 0$, которой принадлежат точки 4, 5 и 6. Для того чтобы исправить эту неточность, нужно немного сдвинуть ключевые точки. К примеру, точка z_1 не должна располагаться точно в $(0, b)$; мы должны внести такую поправку, чтобы верхний край пера находился в точке $(0, b)$, когда его центр находится в точке z_1. Это условие для трех верхних точек можно выразить уравнением

$$top\ z_1 = (0, b); \qquad top\ z_2 = (a, b); \qquad top\ z_3 = (2a, b);$$

точно так же, сдвиг точек 4, 5 и 6 определяется из уравнений

$$bot\ z_4 = (0, 0); \qquad bot\ z_5 = (a, 0); \qquad bot\ z_6 = (2a, 0).$$

Полученный в результате сплющенный символ имеет вид



Здесь он показан в двух версиях: исходной — '⋊' и утолщенной — '⋉'.

▸ **Упражнение 2.5**
Десять утолщенных гексов образуют '⋉⋉⋉⋉⋉⋉⋉⋉⋉⋉'. Отметим, что смежные символы частично накладываются друг на друга. Это объясняется тем, что каждый символ имеет ширину $2a$, и поэтому точка 3 одного символа совпадает с точкой 1 следующего символа. Допустим, требуется, чтобы символы целиком содержались в прямоугольных рамках ширины $2a$ так, чтобы смежные символы лишь слегка соприкасались (⋉⋉⋉⋉⋉⋉⋉⋉⋉⋉). Объясните, как для этого нужно изменить приведенные выше уравнения, определяющие положения точек, имея в виду, что в METAFONT определены операции '*lft*' и '*rt*', аналогичные '*top*' и '*bot*' (обозначающие, соответственно, левый и правый край пера).

Пары координат можно представлять себе не только как точки, но и как векторы или "смещения". К примеру, $(15, 8)$ можно рассматривать как команду перейти на 15 пикселей вправо и на 8 пикселей вверх. Тогда точка $(15, 8)$ — это положение, в которое мы попадем, выполнив команду $(15, 8)$, если в начальный момент мы находились в точке отсчета. В такой интерпретации можно рассматривать операцию сложения векторов: смещение на вектор $(15, 8)$, а затем — на вектор $(7, -3)$ даст тот же результат, что и смещение на $(15, 8) + (7, -3) = (15 + 7, 8 - 3) = (22, 5)$. Сумма двух векторов $z_1 = (x_1, y_1)$ и $z_2 = (x_2, y_2)$ есть вектор $z_1 + z_2 = (x_1 + x_2, y_1 + y_2)$, получаемый в результате сложения компонент x и y по отдельности. Этот вектор представляет результат последовательного смещения на векторы z_1 и z_2. Иначе говоря, $z_1 + z_2$ представляет точку, в которую мы попадем, сместившись из точки z_1 на вектор z_2.

▸ **Упражнение 2.6**
Какой из четырех базисных векторов $(0, 1)$, $(1, 0)$, $(0, -1)$ и $(-1, 0)$ соответствует смещению на один пиксель (a) вправо? (b) влево? (c) вниз? (d) вверх?

Векторы можно не только складывать, но и вычитать. Разность $z_1 - z_2$ есть просто $(x_1 - x_2, y_1 - y_2)$. Более того, естественным образом вводится операция умножения вектора на произвольное число c. Произведение вектора (x, y) и числа c, которое записывается как $c(x, y)$, есть вектор (cx, cy). Таким образом, например, значение $2z = 2(x, y) = (2x, 2y)$ оказывается равным $z + z$. В особом случае, когда $c = -1$, мы пишем $-(x, y) = (-x, -y)$.

Мы подошли к важному понятию, в основе которого лежит тот факт, что вычитание есть операция, обратная сложению. *Если z_1 и z_2 — точки, то $z_2 - z_1$ есть вектор, соответствующий переходу из z_1 в z_2.* Это объясняется просто: $z_2 - z_1$ — это как раз то, что следует прибавить к z_1, чтобы получить z_2, то есть $z_1 + (z_2 - z_1) = z_2$. Мы назовем это правило *принципом вычитания векторов*. Он часто применяется, когда разработчик хочет указать направление перехода из одной точки в другую и/или расстояние между точками.

В системе METAFONT для отражения соотношений между точками часто используется еще одна идея. Предположим, что, стартовав из точки z_1, мы начинаем движение по направлению к точке z_2, но проходим лишь часть пути. Для такого смещения существует специальное обозначение, использующее квадратные скобки:

$\frac{1}{3}[z_1, z_2]$ — одна треть расстояния между z_1 и z_2;

$\frac{1}{2}[z_1, z_2]$ — половина расстояния между z_1 и z_2;

$.8[z_1, z_2]$ — восемь десятых от расстояния между z_1 и z_2.

В общем случае $t[z_1, z_2]$ обозначает точку, которая лежит на расстоянии, составляющем часть t от всего пути между z_1 и z_2. Формально эту операцию мы назовем *помещением в промежуточное положение* между точками, а неформально — функцией "часть от расстояния". Если значение t изменяется от 0 до 1, то $t[z_1,\ z_2]$ пробегает точки на прямой между z_1 и z_2. В соответствии с принципом вычитания векторов, для того чтобы проделать весь путь от z_1 до z_2, мы должны сместиться на $z_2 - z_1$. Следовательно, часть t от расстояния между ними есть

$$t[z_1, z_2] \ = \ z_1 + t(z_2 - z_1).$$

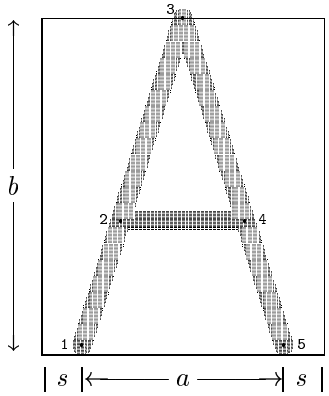
Эта формула позволяет найти $t[z_1, z_2]$ для любых заданных значений t, z_1 и z_2. Но в METAFONT эта формула встроена изначально, так что мы можем, не задумываясь, пользоваться нашим обозначением с квадратными скобками.

Вернемся к шести точкам из первого примера и предположим, что нужно указать точку, лежащую на $2/5$ части пути между $z_2 = (100, 100)$ и $z_6 = (200, 0)$. В METAFONT это записывается как $.4[z_2, z_6]$. Если же нам потребуется вычислить координаты в явном виде, то мы всегда можем вывести их из общей формулы: $z_2 + .4(z_6 - z_2) = (100, 100) + .4\big((200, 0) - (100, 100)\big) = (100, 100) + .4(100, -100) = (100, 100) + (40, -40) = (140, 60)$.

▸ **Упражнение 2.7**
Верно ли, что $(-3, 5)$ — это вектор направления из $(5, -2)$ в $(2, 3)$?

▸ **Упражнение 2.8**
Если формула '$0[z_1, z_2]$' имеет смысл, объясните его. Что означает '$1[z_1, z_2]$'? А '$2[z_1, z_2]$'? А '$(-.5)[z_1, z_2]$'?

▸ **Упражнение 2.9**
Верно ли с точки зрения математики, что: (a) $\frac{1}{2}[z_1, z_2] = \frac{1}{2}(z_1 + z_2)$; (b) $\frac{1}{3}[z_1, z_2] = \frac{1}{3}z_1 + \frac{2}{3}z_2$; (c) $t[z_1, z_2] = (1 - t)[z_2, z_1]$.



В заключение главы попробуем определить при помощи операции помещения в промежуточное положение пять ключевых точек буквы 'A', изображенной в увеличенном виде на рисунке справа. Точки 1 и 5 должны быть расположены на расстоянии a друг от друга, а точка 3 должна находиться на b пикселей выше базовой линии. Значения a и b установлены заранее в соответствии с методом, который мы опишем позже, и точно так же установлено значение параметра s, определяющего расстояние по горизонтали от точек 1 и 5 до края рамки. Мы будем предполагать, что точная высота, на которой должна находиться перекладина, нам неизвестна. Точка 2 должна находиться на прямой линии, соединяющей точки 1 и 3, а точка 4 — в соответствующем положении между точками 5 и 6. Но прежде чем определиться с выбором, мы должны проверить несколько возможных вариантов.

Ширина символа составляет $s + a + s$, и положение точек z_1 и z_5 можно задать формулами

$$bot\ z_1 = (s, 0); \qquad z_5 = z_1 + (a, 0).$$

Это можно сделать и другими способами, но данные формулы ясно выражают наше желание, чтобы нижний край пера касался базовой линии в s пикселях справа от точки отсчета, когда само перо находится в точке z_1, а точка z_5 находилась на a пикселей правее точки z_1. Мы можем выразить это формулой

$$z_3 = \bigl(\tfrac{1}{2}[x_1, x_5], b\bigr).$$

То есть, координата x точки 3 должна быть средним между координатами x точек 1 и 5, а y_3 должно быть равно b. Наконец, укажем

$$z_2 = alpha[z_1, z_3]; \qquad z_4 = alpha[z_5, z_3],$$

где параметр $alpha$ есть число между 0 и 1, которое определяет положение перемычки и которое задается позже. Когда значение $alpha$ уже задано, мы можем написать

$$\textbf{draw}\ z_1 \mathinner{..} z_3; \qquad \textbf{draw}\ z_3 \mathinner{..} z_5; \qquad \textbf{draw}\ z_2 \mathinner{..} z_4.$$

Если $a = 150$, $b = 250$, $s = 30$, и $alpha$ будет меняться от 0,2 до 0,5 с шагом 0,05, то METAFONT нарисует символы 'AAAAAAA'. В иллюстрации на предыдущей странице $alpha = (3 - \sqrt{5})/2 \approx 0.38197$; при таком значении отношение высот верхней и нижней частей фигуры составляет $(\sqrt{5} - 1)/2 \approx 0.61803$ — число, называемое в классической греческой математике золотым сечением.

(Вы уверены, что вам следует читать этот абзац? Знак "опасный поворот" предупреждает, что здесь содержится материал, который должен быть пропущен при первом чтении, а может, и при втором. В таких абзацах, требующих от читателя особого внимания, иногда встречаются понятия, которые объясняются только в последующих главах.)

▸ **Упражнение 2.10**

Почему лучше определить z_3 как $(\frac{1}{2}[x_1, x_5], b)$ и не выводить точные координаты точки $z_3 = (s + \frac{1}{2}a,\ b)$, которые можно определить из других уравнений.

Упражнение 2.11

Пусть точки z_1, z_3 и z_5 — те же, что и выше. Как нужно определить z_2 и z_4, чтобы одновременно выполнялись следующие условия:

- линия из z_2 в z_4 поднимается вверх под углом в $20°$;
- координата y середины этой линии составляет $2/3$ от разности y_3 и y_1;
- z_2 и z_4 расположены, соответственно, на линиях $z_1 \,..\, z_3$ и $z_3 \,..\, z_5$.

(Справившись с этим упражнением, вы получите букву 'A'.)

Достигая сферы математики, мы попадаем в окружение процессов, которые кое-кому кажутся самыми "бесчеловечными" и далекими от поэзии. Но именно здесь художник может дать самую полную свободу своему воображению.

— ХАЙВЛОК ЭЛЛИС, *Танец жизни* (1923)

Для тех, кто жил в одном из современных американских городов (за исключением Бостона), по крайней мере одна из идей, лежащих в основе декартовой аналитической геометрии, покажется до смешного простой. И все же математикам потребовалось целых две тысячи лет, чтобы додуматься до такой простой вещи.

— ЭРИК ТЕМПЛ БЕЛЛ, *Математика: королева и слуга науки* (1951)