УДК 681.142.2

М. А. БЕРЕЗОВСКИЙ, А. К. ЯБЛОНСКИЙ

(Москва)

Введение. Проблема синтеза реалистичных полутоновых изображений трехмерных объектов на основе твердотельных моделей является весьма актуальной в целом ряде приложений машинной графики, в частности при создании машиностроительных САПР [1]. Большой объем вычислений и требование интерактивности приводят к необходимости включения в состав рабочей станции специализированных аппаратных средств, получивших название «геометрический процессор» [2]. По-видимому, основные недостатки этого подхода — сложность и дороговизна графической подсистемы САПР, поскольку геометрический процессор аппаратно ориентирован только на графические вычисления. При этом известно, что в САПР присутствуют и другие вычислительно емкие компоненты такие, как прочностное моделирование, которые для своего эффективного решения также нуждаются в специализированных средствах.

В связи с этим нами в рамках проекта СИГМА* предпринята реализация подсистемы визуализации на базе сравнительно универсального комплекса «основная ЭВМ + векторный процессор (ВП)». Такие комплексы вследствие оптимального соотношения стоимость/производительность являются весьма популярными в области обработки сигналов и численного моделирования [3], в том числе прочностного моделирования методом конечных элементов [4]. Однако показатель стоимость/производительность чрезвычайно важен и для систем САПР вследствие предъявляемых к ним требований широкой тиражируемости и интерактивности (~10 с на синтез изображения).

Исследования параллелизма в графических вычислениях проводились рядом авторов (см., например, [5—8]) и показали обнадеживающие результаты. Мы же видим свою задачу в разработке параллельных алгоритмов интерактивной полутоновой графики и других вычислительно емких подсистем машиностроительной САПР для эффективного и многофункционального вычислительного комплекса. В частности, в графической подсистеме ВП выполняет значительную часть функций геометрического процессора.

Базовый вычислительный комплекс в нашем случае включает в себя ЕС ЭВМ в качестве основной и ВП типа ЕС 2706 [9]. В качестве устройства отображения был использован цветной растровый видеотерминал СВИТ [10]. Программирование на языке высокого уровня позволяет перенести разработанный пакет программ на комплекс «СМ-ЭВМ + ВП типа А-12» [11]**.

Для геометрического моделирования в САПР характерны два класса форм, покрывающих почти весь спектр приложений: булевские комбинации геометрических примитивов*** и скульптурные поверхности. В статье предлагается разработанный и реализованный в СИГМе алгоритм визуализации объектов первого типа. Для скульптурных поверхностей работа находится в стадии разработки. В обоих алгоритмах основное внимание уделяется эффективности распараллеливания на ВП с

---

* СИГМА — система интерактивного геометрического моделирования для автоматизированного проектирования.

** Векторные процессоры ЕС 2706 и А-12 архитектурно близки и программно совместимы с процессорами FPS AP-190L и AP-120B соответственно.

*** Для этого класса тел принят термин CSG (Constructive Solid Geometry).

целью добиться производительности, необходимой для интерактивного режима.

1. Описание алгоритма. Параллелизм, присущий архитектуре ВП, эффективно реализуется в векторных и конвейерных операциях. Первые ционального устройств, задействованные в библиотечной или специализированной для данного приложения подпрограмме, загружаются полностью и в течение длительного времени, что обеспечивает их наивысшую производительность.

Далее ПО представлены в нотации, в которой те индексы векторных операндов, по которым допускается параллельная обработка, заменены символом #. Индексное множество, на котором выполняется поэлементно независимая обработка, указывается в фигурных скобках справа от ПО или группы ПО, если эти множества для всех ПО группы совпадают, например:

$$A(\#)=B(\#)+C(\#),$$
$$D(\#)=\max(D(\#),\ A(\#))\ \ \{1:n\}.$$

Семантика векторных и скалярно-векторных операций понимается традиционно, т. е. как поэлементное выполнение.

1.1. Обзор основных этапов. В основе визуализации CSG-тел лежит подход, известный под названием «трассировка луча» [12, 13]. Изображение строится непосредственно по булевскому описанию, минуя вычисление модели поверхности. Видимые точки объекта определяются с помощью лучей, проходящих через элементы экранной матрицы вдоль направления наблюдения. Работа алгоритма состоит из следующих этапов.

А. Выполнение преобразования проецирования из системы координат объекта на экран. После этого уравнения лучей принимают вид $x=x_0$, $y=y_0$, что упрощает вычисления.

Б. Нахождение точки пересечения лучей с поверхностями примитивов. Для векторизации алгоритма параллельно обрабатываются 256 лучей, проходящих через фрагмент экранной матрицы размером 16×16 *. По окончании этого этапа лучи, не пересекающие ни одного примитива, должны быть отброшены.

В. Выделение точки поверхности результирующего тела с выполнением одномерных булевских операций над полученными на этапе 2 отрезками лучей. Побочным результатом является упорядоченность найденных точек по величине z-координаты.

Г. Вычисление окраски поверхности и запись в буфер изображения в порядке убывания z-координат. Этот способ не оптимален в том смысле, что окраска вычисляется как для видимых, так и для затираемых точек. Однако ему было отдано предпочтение, поскольку он пригоден для визуализации полупрозрачных поверхностей.

1.2. Модель объекта. Предлагаемая модель CSG-объекта, описываемая в данном разделе, разрабатывалась с целью полного распараллеливания графического алгоритма.

---

* С точки зрения векторизации фрагмент должен содержать как можно больше элементов. Однако его размер должен быть невелик по сравнению с примитивом, чтобы число лучей, проходящих мимо, было мало. Матрица 16×16 является приемлемым компромиссом, поскольку средний размер примитива обычно занимает от 1/8 до 1/2 стороны экрана, что при разрешении 256×256 составляет 32—128 элементов.

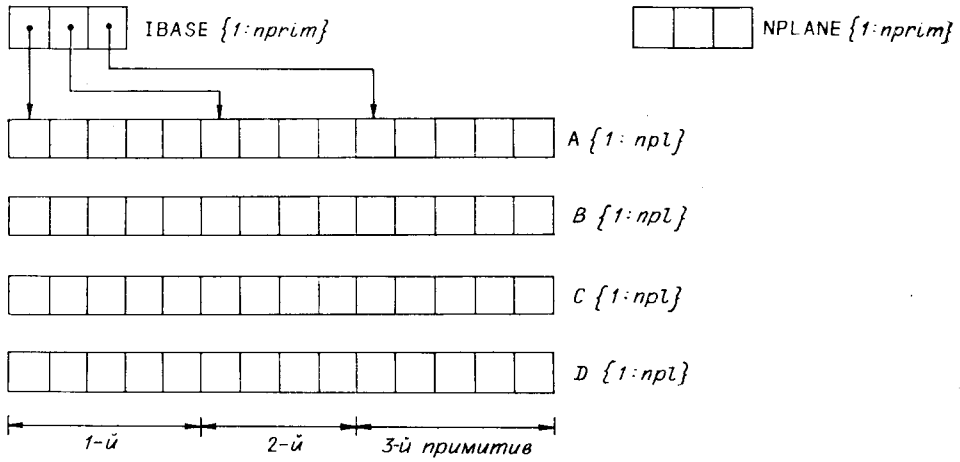

Рис. 1

Нами была использована разновидность CSG, предложенная в [14]. В ней объект S представлен объединением n примитивов P_i, $1 \leqslant i \leqslant n$:

$$S = \bigcup_{i=1}^{n} P_i = (((\ldots(\varnothing \cup P_1) \cup P_2)\ldots) \cup P_{n-1}) \cup P_n. \qquad (1)$$

В настоящее время реализовано четыре типа параметризованных примитивов: параллелепипед, конус, цилиндр и сфера. Для реализации булевского вычитания каждому P_i приписана мода, принимающая два значения: «+» и «—». Соответственно этому будем считать, что примитив имеет положительный или отрицательный знак.

Объем, занимаемый примитивом, описывается в виде пересечения плоских полупространств [15]. Полупространство есть множество точек, удовлетворяющих неравенству

$$ax + by + cz + d \leqslant 0,$$

где x, y, z — координаты точки; a, b, c, d — вещественные числа.

Кроме коэффициентов полупространств, с каждым примитивом связана объемлющая его сфера, используемая в местах перекрытия с фрагментом экрана.

Описываемая модель хранится в памяти ВП в виде одномерных массивов, изображенных на рис. 1. В массивах A, B, C и D находятся коэффициенты полупространств. IBASE(i) содержит базовый индекс, начиная с которого записаны коэффициенты i-го примитива, NPLANE(i) есть число этих коэффициентов, nprim — число примитивов, входящих в объект, а npl — общее число полупространств.

Кроме этих данных, модель занимает еще восемь массивов, не показанных на рис. 1. В XS, YS, ZS, RS {1 : nprim) находятся координаты центров и радиусы объемлющих сфер. В массиве VMODE {1 : nprim} содержатся моды, а в R, G, B{1 : nprim} — компоненты цвета примитивов.

1.3. Преобразование проецирования. Ортогональное проецирование (этап А) задается матрицей вращения и масштабирования ROT {1 : 3, 1 : 3}, а также переносом (t_x, t_y, t_z).

Исходная четверка (a, b, c, d) переходит в новые коэффициенты (a', b', c', d') следующим образом:

$$(a', b', c')^T = ROT * (a, b, c)^T; \qquad (2)$$

$$d' = d - ((a', b', c'), (t_x, t_y, t_z)). \qquad (3)$$

Первая формула может быть интерпретирована геометрически как вращение нормали, а вторая — как вычисление расстояния от плоскости до нового начала координат.

В терминах ПО преобразование (2) реализовано как обращение к стандартной подпрограмме умножения матрицы на последовательность столбцов.

Формула (3) принимает следующий вид:

$$D(\#)=D(\#)-t_x * A(\#)-t_y * B(\#)-t_z * C(\#)\ \{1:\mathrm{npl}\}.$$

1.4. Пересечение с поверхностями примитивов. Поскольку изображение строится в виде последовательности фрагментов, этапы Б — Г повторяются для каждого из них.

На этапе Б производится перебор примитивов P_i, входящих в объект. Если фрагмент не перекрывается объемлющей сферой примитива, перебор продолжается дальше. В противном случае для каждого луча вычисляются z-координаты точек пересечения с поверхностью примитива.

z-координата пересечения луча ($x=x_0$, $y=y_0$) с j-й плоскостью $a_jx+b_jy+c_jz+d_j=0$ находится как $z_j=-(a_jx_0+b_jy_0+d_j)/c_j$ при условии, что $c_j\neq 0$.

Полупространства с положительными коэффициентами при z (их принято называть передними) отсекают на прямой луча область $[z_{\mathrm{in}}, +\infty)$, где

$$z_{\mathrm{in}}=\max\left(z_{j_k}\right),\ c_{j_k}>0. \tag{4}$$

Аналогично для полупространств с $c_{i_m}<0$ (задних) отсекаемая область есть $(-\infty, z_{\mathrm{out}}]$, где

$$z_{\mathrm{out}}=\min\left(z_{i_m}\right). \tag{5}$$

У областей (4) и (5) будут общие точки, если $z_{\mathrm{in}}\leqslant z_{\mathrm{out}}$. Для того чтобы имело место пересечение поверхности примитива, неравенство должно быть строгим: $z_{\mathrm{in}}<z_{\mathrm{out}}$. Тогда $(x_0, y_0, z_{\mathrm{in}})$ и $(x_0, y_0, z_{\mathrm{out}})$ есть искомый результат.

Особую ситуацию представляют плоскости с нулевыми коэффициентами при $z: ax+by+d=0$. Точки пересечения отсутствуют, и луч целиком находится либо внутри полупространства, либо снаружи, либо на границе. Соответственно в первом случае $ax_0+by_0+d<0$, а во втором — $ax_0+by_0+d>0$, z_{in} и z_{out} вычисляются как последовательное приближение путем перебора всех полупространств примитива.

Приведем пример ПО для переднего полупространства:

$$z(\#)=(a * X(\#)+b * Y(\#)+d)/c;$$
$$\mathrm{ZIN}(\#)=\max(\mathrm{ZIN}(\#),\ z(\#))\ \{1:256\}.$$

Здесь векторы X и Y содержат x- и y-координаты точек фрагмента. Остальные типы полупространств обрабатываются аналогичным образом.

После того как вычислены векторы ZIN и ZOUT, содержащие z_{in} и z_{out}, происходит отсев лучей, проходящих мимо примитива. Для этого вычисляется разность ZOUT — ZIN и данные, соответствующие положительным элементам результата, с помощью ПО записываются в выходные массивы. Затем берется следующий примитив объекта и цикл повторяется, начиная с теста перекрытия. По окончании цикла информация о точках пересечения лучей с примитивами собрана в векторах, каждый элемент которых содержит информацию об одной точке:

IFRAG(i) — порядковый номер луча, пересекающего поверхность примитива;

ZFRAG(i) — z-координата точки пересечения;

PRTAG(i) — индекс пересекаемого примитива;

PLTAG(i) — индекс пересекаемой плоскости;

PLSIGN(i) — +1, если плоскость передняя; −1, если плоскость задняя.

Число элементов записано в переменной nfrag.

1.5. Выделение поверхности объекта. Свяжем с каждым примитивом объекта P_i ($1\leqslant i\leqslant$ nprim) функцию $\Phi_i(x, y, z)$ такую, что:

1) Φ_i определена всюду, кроме точек поверхности i-го примитива;

2) $\Phi_i(x, y, z)\begin{cases}\Psi_i = 2^i \operatorname{sign} V_i & \text{во внутренних точках } P_i;\\ 0 & \text{в остальных точках,}\end{cases}$

где V_i — мода примитива.

Введем функцию Φ, определенную во всех точках пространства, кроме поверхностей примитивов, и равную

$$\Phi(x, y, z) = \sum^{\text{nprim}} \Phi_i(x, y, z). \qquad (6)$$

принадлежат результирующему объекту. G и H есть места пересечения луча с поверхностью тела, так как в любой их окрестности найдутся z' и z'' такие, что $\Phi(x_0, y_0, z') > 0$, а $\Phi(x_0, y_0, z'') \leqslant 0$. Для проверки этих условий можно было бы выбрать тестовые точки и найти в них сумму (6). Однако существует более экономичный способ.

Возьмем точку E. Поскольку она имеет наименьшую z-координату, все примитивы находятся справа от нее и предел значений функции слева $\Phi^-(E) = 0$. Для предела справа справедливо

$$\Phi^+(E) = \Phi^-(E) + \Psi_2 = 0 + (2^2(-1)) = -4.$$

Поскольку в точке E имеет место пересечение со вторым примитивом, то Φ_2 изменяет свое значение.

Так как на интервале EF функция Φ постоянна, $\Phi^-(F) = \Phi^+(E) = -4$.

В точке F луч входит в первый примитив и изменяется значение Φ_1:

$$\Phi^+(F) = \Phi^-(F) + \Psi_1 = -2.$$

Далее, для G

$$\Phi^-(G) = \Phi^+(F) = -2;\ \Phi^+(G) = \Phi^-(G) - \Psi_2 = +2,$$

так как луч выходит из P_2 и Φ_2 вместо -4 становится нулем. И наконец, в H

$$\Phi^-(H) = \Phi^+(G) = +2;\ \Phi^+(H) = \Phi^-(H) - \Psi_1 = 0.$$

Таким образом, значения Φ вычислялись рекуррентно при продвижении вдоль луча в направлении возрастания z. Для этого точки E, F, G и H должны быть упорядочены (отсортированы) по величине z. С этой целью был запрограммирован алгоритм «сортировки подсчетом» [16]. Отсутствие условных операторов и поэлементный характер действий делают его идеальным с точки зрения распараллеливания.

По завершении сортировки производится вычисление пределов функции Φ с помощью процедуры, описанной в этом разделе. Данная процедура также обладает свойством независимости обработки каждой точки, что обеспечивает эффективность реализации на архитектуре ВП. Результаты этого этапа записываются в перечисляемые ниже векторы, каждый элемент которых содержит информацию о точке поверхности объекта:

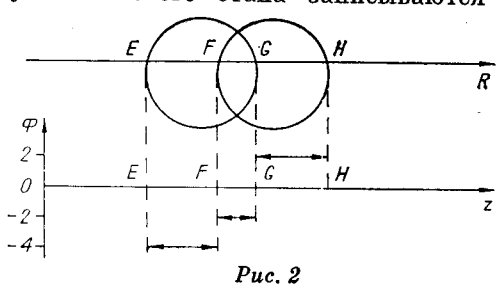

Рис. 2

ISOL(i) — номер луча, на котором лежит точка;

PLSOL(i) — индекс пересекаемой плоскости;

PRSOL(i) — индекс примитива.

Переменная nsolid содержит число элементов в каждом массиве.

поверхности и компонент нормалей. Эта структура данных близка к полученной нами в результате предыдущего этапа, так как по индексам примитивов можно извлечь цветовые характеристики из массивов R, G и B, а нормали — из массивов A, B и C. Основное отличие состоит в том, что в [18] длина операндов ПО равна горизонтальному разрешению экрана и каждый элемент массивов соответствует пикселу изображения. При таком подходе большая часть времени идет на вычисление заранее известной фоновой интенсивности. В изображениях же, типичных для САПР, фон занимает не менее 2/3 площади. Кроме того, в подходе, развитом в [18], невозможно изображение полупрозрачных тел.

В нашем случае наличие точек поверхности, не распределенных на экранной матрице, дает возможность выбрать два пути. В первом, применяемом для непрозрачных объектов, сначала определяются окончательно видимые точки.

Это осуществляется с помощью следующей ПО:

$$\begin{aligned} &\mathrm{VIS}(\#) = \mathrm{FLAG}(\mathrm{ISOL}(\#)), \\ &\mathrm{FLAG}(\mathrm{ISOL}(\#)) = 0 \ \{1 : \mathrm{nsolid}\} \end{aligned} \tag{7}$$

(предполагается, что первоначально вектор FLAG заполнен единицами). После этого в массиве VIS видимые точки отмечены единицами, а остальные — нулями. Оперируя данными, соответствующими единичным элементам VIS, окраска вычисляется только для видимых точек. Объем вычислений при этом пропорционален содержательной части изображения.

Для полупрозрачных тел окраска определяется для всех точек, после чего производится запись в буфер фрагмента линейной комбинации окрасок поверхностей переднего и последующих планов, что дает желаемый эффект [18].

1.7. Оценка объема вычислений. Для краткости изложения ограничимся анализом вычисления z-координат пересечений (этап Б), т. е. самого внутреннего цикла алгоритма. Объем остальных вычислений для проведенных тестов составил не более 30% внутреннего цикла.

Обозначим: N_p — число примитивов объекта; H_p — характерный размер примитива в элементах изображения; h — среднее число полупространств в примитиве; f — размер фрагмента; m, n — горизонтальное и вертикальное разрешения экрана; t — время обработки одного элемента массива с помощью ПО. Тогда время, затрачиваемое на пиксел изображения, составит

$$T = n_p h t, \tag{8}$$

где n_p — число примитивов, прошедших тест перекрытия с фрагментом.

Для n_p примем следующее приближение, смысл которого поясняется на рис. 3:

$$n_p = N_p \frac{(f + 2H_p)^2}{mn} \simeq 4N_p \frac{H_p^2}{mn}.$$

Предполагается, что f невелико по сравнению с H_p. Подставляя в (8), получаем

$$T = 4N_p \frac{H_p^2}{mn} h t. \tag{9}$$

В имеющейся реализации $t \simeq 0{,}5$ мкс. Считая, что H_p составляет 1/5 стороны экрана и полагая h равным 30, получаем для объекта, состоящего из 30 примитивов, $T = 72$ мкс. Для матрицы 256×256 общее время будет около 5 с.

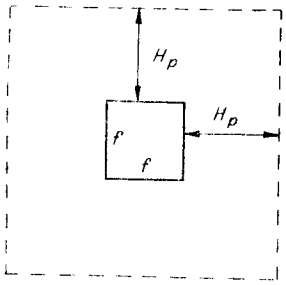

Рис. 3

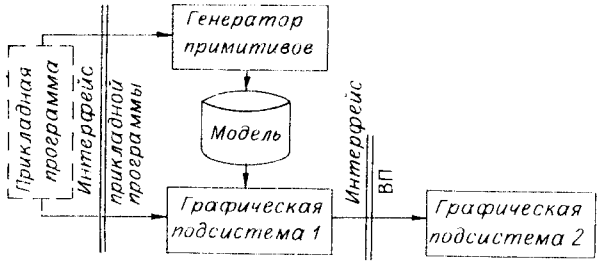

Рис. 4

2. Усовершенствование алгоритма. В этом разделе описываются модификации, которые планируется реализовать в будущем для улучшения различных аспектов работы алгоритма.

2.1. Сглаживание окраски. Плоскостная аппроксимация криволинейных поверхностей приводит к разрывности функции окраски на стыках полупространств. Для полигональных моделей известен способ «сглаживания», использующий вместо истинных нормалей приближения к нормалям аппроксимируемой поверхности [17]. Для конических, цилиндрических и сферических поверхностей компоненты нормалей есть линейные функции координат и, следовательно, легко могут быть вычислены, так как координаты точек поверхности известны во время работы алгоритма. Описания примитивов должны быть дополнены соответствующими коэффициентами. Таким образом, можно добиться сокращения времени вычислений и объема памяти за счет уменьшения числа используемых плоскостей.

2.2. Ограниченная модификация. При интерактивной работе часто изменяется лишь небольшая часть изображения, как, например, при добавлении нового примитива. Наличие объемлющей сферы позволяет локализовать область модификации и ограничить вычисления только перекрывающими ее фрагментами по аналогии с методом, предложенным в [13].

Заключение. Программная архитектура системы представлена на рис. 4. Прикладная программа формирует описание объекта с помощью генератора примитивов, каждое обращение к которому дополняет модель новым примитивом. Параметры наблюдения и характеристики освещения задаются путем вызова подпрограмм графической подсистемы, имеющей резидентные части как в ВП, так и в основной ЭВМ. После того как задана вся необходимая информация, данные передаются в ВП, где и происходит пофрагментное вычисление изображения.

Организующая часть системы реализована на языке Фортран основной ЭВМ. Графическая подсистема *2* выполнена в основном на базе подпрограмм стандартной векторной библиотеки ВП. С их помощью построено большинство ПО, используемых алгоритмом. Остальные ПО, а именно (7), реализованы на ВП специализированными ассемблерными процедурами.

Эксперименты с системой подтвердили правильность оценки (9). Изображения тестовых объектов приведены в [20].

ЛИТЕРАТУРА

1. **Borrell J.** The solid modelling marketplace // Computer Graphics World.— 1982.— N 11.— P. 45.
2. **Clark J. H.** The geometry engine: a VLSI geometry system for graphics // Computer Graphics.— 1982.— V. 16.— P. 127.
3. **Родрига Г.** Параллельные вычисления.— М.: Наука, 1986.
4. **Hindin J. H.** Fifth-generation computing: dedicated software is the key // Computer Design.— 1984.— N 9.— P. 150.
5. **Kaplan M., Greenberg D. P.** Parallel processing techniques for hidden surface removal // Computer Graphics.— 1979.— V. 13.— P. 300.

6. **Fiume E., Fournier A., Rudolph L.** A parallel scan-conversion algorithm with anti-aliasing for a general purpose ultracomputer // Computer Graphics.— 1983.— V. 17.— P. 141.
7. **Fuchs H. e. a.** Fast sphere, shadows, textures, transparencies and image enhancements in pixel-planes // Computer Graphics.— 1985.— V. 19.— P. 111.
8. **Sato H. e. a.** Fast image generation of constructive solid geometry using cellular array processor // Computer Graphics.— 1985.— V. 19.— P. 95.
9. **Марков С., Лазаров В.** Специализированные высокопроизводительные вычислительные системы // Вычислительная техника социалистических стран: Сб. статей.— М.: Финансы и статистика, 1986.— Вып. 19.
10. **Чесалин Л. С. и др.** Самостоятельный видеоинформационный терминал СВИТ.— М., 1982.— (Препринт/АН СССР, ИКИ; 721).
11. **Бродский И. И и др.** Высокопроизводительный периферийный векторный процессор А-12 // Вопросы кибернетики.— М.: Научный совет по комплексной проблеме «Кибернетика» АН СССР. 1984. № ВК-104.
12. **Goldstein R. A., Nagel R.** 3-D visual simulation // Simulation.— 1971.— V. 16.— P. 25.
13. **Roth S. D.** Ray casting for modelling solids // Computer Graphics and Image Processing.— 1982.— V. 18.— P. 109.
14. **Okino N., Kakazu Y., Kubo H.** Theories for graphics processors in TIPS-1 // Computers and Graphics.— 1983.— V. 7.— P. 243.
15. **Requicha A.** Representations for rigid solids: theory, methods and systems // ACM Computing Surveys.— 1980.— V. 12.— P. 437.
16. **Кнут Д. Е.** Искусство программирования для ЭВМ.— Т. 3: Сортировка и поиск.— М.: Мир, 1976.
17. **Phong B. T.** Illumination for computer-generated pictures // Commun. of the ACM.— 1975.— V. 18.— P. 311.
18. **Curington I. J.** A normal buffer vectorized surface shading model // Eurographics'85 Proc.— Amsterdam: North-Holland, 1985.
19. **Newman W. M., Sproull R. F.** Principles of interactive computer graphics.— N. Y.: McGraw-Hill, 1979.
20. **Алексеенко М. М., Березовский М. А., Свирин Б. Н. и др.** Интерактивная система геометрического моделирования для СМ ЭВМ и векторного процессора // Микропроцессорные средства и системы.— 1987.— № 5.

Поступила в редакцию 21 апреля 1987 г.

УДК 681.3

Н. Л. КУЦЕВИЧ, А. Я. ОЛЕЙНИКОВ, Е. В. ПАНКРАЦ, В. А. ТИМОФЕЕВ
(Москва)

СРЕДСТВА ПРОГРАММИРОВАНИЯ ДЛЯ МОДУЛЯ ПРИБОРНОГО ИНТЕРФЕЙСА

В настоящее время в качестве основного средства для сопряжения ЭВМ с измерительным и управляющим оборудованием и построения систем автоматизации эксперимента (САЭ) используется аппаратура в стандарте КАМАК. Вместе с тем отмечается тенденция к построению относительно простых САЭ, в которых число измерительных приборов не превышает 10—15 и суммарная скорость обмена 100—200 Кбайт/с, на основе приборного интерфейса (МЭК 625.1, ГОСТ 26.003—80, «Канал общего пользования» — КОП) [1].

Для построения САЭ может использоваться непосредственное сопряжение магистрали КОП с ЭВМ. Наряду с этим наличие у пользователей аппаратуры в стандарте КАМАК и продолжающийся широкий промышленный выпуск аппаратуры и измерительно-вычислительных комплексов на ее основе делают целесообразным создание модулей КАМАК, выполняющих функции контроллера магистрали приборного интерфейса [2]. Один из таких модулей — модуль приборного интерфейса (МПИ) [3], который выпускается Экспериментальным заводом научного приборостроения АН СССР.