РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ
А В Т О М Е Т Р И Я
№ 5 1993

УДК 681.3.06

А. В. Романовский
(Новосибирск)

VML — ЯЗЫК ОПИСАНИЯ ГЕОМЕТРИЧЕСКИХ МОДЕЛЕЙ ТРЕХМЕРНЫХ СЦЕН ДЛЯ СИНТЕЗИРУЮЩИХ СИСТЕМ ВИЗУАЛИЗАЦИИ

Описывается язык геометрического моделирования поверхностей трехмерных объектов, предоставляющий средства конструирования поверхностей вращения, сплайн-поверхностей, выпуклых оболочек и других объектов-примитивов, обеспечивающий конструирование объектов путем применения теоретико-множественных операций конструктивной геометрии: объединения, пересечения, вычитания.

Введение. VML применяется в системе подготовки баз данных [1, 2] для синтезирующих систем визуализации (ССВ) [2, 3] и является «неортогональным дополнением» к языку SDL [4]. VML предоставляет средства конструирования поверхностей вращения, сплайн-поверхностей, выпуклых оболочек и других объектов-примитивов, а также обеспечивает конструирование объектов путем применения теоретико-множественных операций конструктивной геометрии: объединения, пересечения, вычитания. Для формального определения языка используется нотация Бэкуса — Наура, как в [4].

Алфавит и словарь языка. Алфавит языка состоит из букв, цифр и прочих графем ASCII, словарь языка (множество лексем) — как в [4].

⟨служебная лексема⟩ : : =
and | AT | DO | ELIF | ELSE | FI | FO | FOR | GO | IF | IS | MADE |
MODEL | neg | not | OBJECT | OD | OF | or | OUT | SI | THEN |
TO | '+' | '–' | '*' | ';' | '(' | ')' | '=' | '[' | ']' | '{' | '}'
| ': =' | '⟨' | '⟨=' | '⟩' | '⟩=' | '/' | ',' ;
⟨имя⟩ : : = ⟨буква⟩ [{⟨буква⟩ | ⟨цифра⟩ | '_'}] ;

Множество имен не пересекается со множеством служебных лексем.

Строковые литералы представляют собой последовательности, возможно, нулевой длины графем алфавита языка, ограниченные двумя парными одиночными или двойными кавычками. Кавычки являются частью литерала.

Примеры строковых литералов:

'test' "exam" '123"8*' "qwerty'" "': ='"
⟨числовой литерал⟩ : : = ⟨число⟩['.'⟨число⟩][⟨масштаб⟩] ;
⟨масштаб⟩ : : = 'e' ['+' | '–']⟨число⟩ ;

Число в ⟨масштабе⟩ задает степень 10. При отсутствии знаков '+','–' подразумевается '+'.

Примеры числовых литералов:

32467 123 41 0 32768
14.01e-09 3.1415926
⟨скобочный комментарий⟩ : : = '(*' [{ ⟨графема ASCII⟩ }] '*)' ;

⟨комментарий до конца строки⟩ : : = '!' [{ ⟨графема ASCII⟩ }] ;

Для сегментации программ используется прагмат включения текста другого файла в данный файл:

⟨прагмат включения текста другого файла в данный файл⟩ : : =
'%include' ⟨имя файла⟩ ;

При компиляции прагмат заменяется текстом из файла, имя которого указано после '%include'.

Типы данных. В VML нет средств для определения типов данных, так как набор встроенных типов данных языка считается достаточным для описания геометрических моделей довольно представительного класса 3-мерных сцен. Встроенные типы данных представлены логическим и числовым типами, типами векторов в трехмерном пространстве, последовательности векторов, типом матриц аффинного преобразования, строковым типом. Для обозначения типов используются имена Logical, Numeral, Vector, Sequence, Matrix, String соответственно.

Множество значений логического типа состоит из истинного и ложного значений, которым соответствуют константы с именами true, false. К данным логического типа применимы одноместная операция логического отрицания, двухместная мультипликативная операция логического ИЛИ и двухместная аддитивная операция логического И, возвращающие результат логического типа. Для обозначения операций используются служебные лексемы not, or, and.

Множество значений числового типа состоит из целых чисел и чисел с плавающей точкой, которым соответствуют числовые литералы из словаря языка. К данным числового типа применимы одноместная операция получения обратного значения (унарный минус), двухместные мультипликативные операции умножения и деления, двухместные аддитивные операции сложения и вычитания, возвращающие результат числового типа. Для обозначения этих операций используются служебные лексемы –, *, /, +, –. К данным числового типа применимы также двухместные операции сравнения с традиционной семантикой ⟩, ⟨, ⟩=, ⟨=, =, ⟨ ⟩, возвращающие результат логического типа.

Множество значений типа векторов в трехмерном пространстве состоит из троек значений числового типа. Первое значение в тройке соответствует координате X вектора, второе значение — координате Y, третье значение — координате Z. Встроена операция конструирования данных типа Vector:

⟨конструктор вектора⟩ : : =
'{' ⟨координата X⟩ ':' ⟨координата Y⟩ ':' ⟨координата Z⟩ '}' ;

К данным типа Vector применимы одноместная операция получения обратного вектора, двухместные мультипликативные операции скалярного и векторного умножения, двухместные аддитивные операции сложения и вычитания, возвращающие результат типа Vector. Для обозначения этих операций используются служебные лексемы –, *, **, +, –.

К операндам типа Vector и Numeral применимы двухместные мультипликативные операции умножения вектора на скаляр и деления вектора на скаляр, возвращающие результат типа Vector. Для обозначения этих операций используются служебные лексемы *, /.

Множество значений типа последовательности векторов состоит из конечных последовательностей элементов типа Vector, включая пустую последовательность. Встроена операция конструирования данных типа Sequence.

⟨конструктор последовательности векторов⟩ : : =
'[' [⟨элемент типа Vector⟩ [{ ',' ⟨элемент типа Vector⟩ }]]']' ;

К данным типа Sequence применима двухместная аддитивная операция слияния последовательностей, возвращающая результат типа Sequence. Для обозначения этой операции используется служебная лексема +.

Множество значений типа матриц аффинного преобразования состоит из аффинных преобразований трехмерного пространства. Конструкторами матриц являются встроенные функции с именами move, reflect, rotX, rotY, rotZ, scale, возвращающие результат типа Matrix.

Функция move имеет один параметр типа Vector и возвращает матрицу перемещения, соответствующего значению параметра.

Функция reflect имеет два параметра типа Vector и возвращает матрицу зеркального отражения относительно плоскости, задаваемой нормалью и точкой, которым соответствуют значения параметров.

Функция rotX имеет один параметр типа Real и возвращает матрицу поворота вокруг оси OX на угол в радианах, значение которого задается параметром.

Функция rotY имеет один параметр типа Real и возвращает матрицу поворота вокруг оси OY на угол в радианах, значение которого задается параметром.

Функция rotZ имеет один параметр типа Real и возвращает матрицу поворота вокруг оси OZ на угол в радианах, значение которого задается параметром.

Функция scale имеет один параметр типа Vector и возвращает матрицу масштабирования по координатным осям в соответствии со значениями компонент параметра.

К данным типа Matrix применима двухместная мультипликативная операция композиции задаваемых операндами аффинных преобразований трехмерного пространства. Операция возвращает результат типа Matrix, и для ее обозначения используется служебная лексема *.

К операндам типа Vector и Matrix применима двухместная мультипликативная операция преобразования вектора, возвращающая результат типа Vector. Для обозначения этой операции используется служебная лексема *.

К операндам типа Sequence и Matrix применима двухместная мультипликативная операция преобразования векторов последовательности, возвращающая результат типа Sequence. Для обозначения этой операции используется служебная лексема *.

Множество значений строкового типа состоит из строк конечной длины, которым соответствуют строковые литералы из словаря языка.

Определение переменных. Переменные определяются до их использования. В определении переменной указывается ее тип данных, имя и, возможно, выражение, в результате исполнения которого переменной присваивается начальное значение.

⟨определение переменных⟩ : : =
⟨имя типа данных⟩
⟨определяемая переменная⟩[{','⟨определяемая переменная⟩}] ;
⟨определяемая переменная⟩ : : = ⟨имя переменной⟩[': ='⟨ выражение⟩] ;

Объекты. Все моделируемые трехмерные тела и поверхности представляются наборами граней. Таким наборам граней в языке соответствуют объекты. Каждый определяемый в VML-программе объект есть либо параметризованный встроенный объект, либо композиция параметризованных встроенных объектов. Объекты определяются до их вызова.

⟨определение объекта⟩ : : =
'OBJECT' ⟨заголовок объекта⟩
'IS'
{⟨определение переменной⟩|⟨определение объекта⟩|⟨оператор⟩}
'SI' ;

⟨заголовок объекта⟩ : : =
⟨имя объекта⟩ [⟨формальные параметры объекта⟩ | ;
⟨формальные параметры объекта⟩ : : =
'('

Ниже приводятся заголовки параметризуемых встроенных объектов, в соответствии с которыми их следует вызывать. Первым параметром во всех объектах является номер цвета объекта, последним — признак гладкости объекта. Если значение последнего параметра равно true, то с гранями объекта ассоциируется информация, которая используется алгоритмом растрирования при генерации изображения для придания стыкам граней визуальной гладкости. При значении параметра, равном false, стыки граней не будут визуально гладкими. Термин «профиль» означает ломаную кривую, задаваемую последовательностью из двух или более несовпадающих векторов. Если первая и последняя точки профиля совпадают, профиль считается замкнутым.

Полигональная аппроксимация поверхности Безье.

```
bezier(Numeral Color;
       Sequence ControlGrid;
       Numeral ControlRows,ControlColumns,SurfaceRows,
       SurfaceColumns;
       Logical Smooth)
```

ControlGrid задает управляющие точки двумерной решетки размером ControlRows * ControlColumns. Поверхность Безье аппроксимируется треугольниками, число которых определяется как 2 * SurfaceRows * SurfaceColumns.

Выпуклая оболочка.

```
convex(Numeral Color;
       Sequence A;
       Logical Smooth)
```

Выпуклая оболочка строится над векторами, содержащимися в последовательности A.

Поверхность вращения.

```
pivot(Numeral Color;
      Sequence A;
      Vector Axis,Origin;
      Numeral Angle,EdgeFaces;
      Logical Smooth)
```

Поверхность является результатом вращения профиля, задаваемого последовательностью A, в 3-мерном пространстве относительно оси, задаваемой вектором Axis и проходящей через Origin, на угол Angle. EdgeFaces задает количество граней для одного ребра профиля при повороте на угол Angle.

Плоская поверхность со сложным контуром.

```
polygon(Numeral Color;
        Sequence A)
```

Поверхность представляет собой полигон, вершины которого соответствуют *векторам* последовательности A.

Профилированная поверхность.

```
profile(Numeral Color;
        Sequence Top,Bottom;
        Logical Smooth)
```

Поверхность определяется двумя профилями Top и Bottom с одинаковым числом векторов в каждом.

Поверхность пирамиды со сложным контуром основания.

```
pyramid(Numeral Color;
        Vector Top;
        Sequence Bottom;
        Logical Smooth)
```

Поверхность соответствует пирамиде с вершиной в точке Top и основанием Bottom, которое может быть произвольным профилем.

Полигональная аппроксимация поверхностей Q-сплайнов.

```
qSpline(Numeral Color;
        Sequence ControlGrid;
        Numeral ControlRows,ControlColumns,
        InRowIntervals,InColumnIntervals;
        Logical Smooth)
```

Поверхность Q-сплайна аппроксимируется треугольниками. Значения параметров ControlGrid, ControlRows, ControlColumns аналогичны bezier. InRowIntervals, InColumnIntervals задают число разбиений интервалов при полигональной аппроксимации по строкам и столбцам управляющей решетки соответственно.

Операторы языка.

```
⟨оператор⟩ : : =
[
⟨оператор присваивания⟩|
⟨условный оператор⟩|
⟨оператор цикла⟩|
⟨оператор выхода из цикла⟩|
⟨оператор задания состава обьекта⟩
];

⟨оператор присваивания⟩ : : = ⟨имя переменной⟩ ': =' ⟨выражение⟩ ;
```

После исполнения оператора значение переменной, чье имя находится слева от ': =', будет равно значению выражения справа от ': ='. Результат выражения должен быть того же типа, что и переменная.

```
⟨условный оператор⟩ : : =
'IF' ⟨вариант⟩ ['ELIF'⟨вариант⟩] ['ELSE'{⟨оператор⟩}] 'FI' ;
⟨вариант⟩ : : = ⟨выражение⟩ 'THEN' {⟨оператор⟩} ;
```

Выражения в вариантах должны вырабатывать результат типа Logical. При исполнении условного оператора выполняются операторы того варианта, значение выражения которого равно true, и исполнение условного оператора завершается. Если значения всех выражений вариантов равны false, то исполняются операторы оборота ELSE, если таковой имеется, и исполнение условного оператора завершается.

⟨оператор цикла⟩ : : =
['FOR'⟨имя переменной цикла⟩':='⟨выражение1⟩]
['TO' ⟨выражение2⟩] 'DO'
{⟨оператор⟩}
'OD' ;

Переменная цикла должна иметь тип Numeral, а выражения должны вырабатывать результат типа Numeral. Операторы, вложенные в оператор цикла, называются телом цикла.

Если обороты FOR и TO присутствуют, то исполнение оператора цикла начинается с вычисления выражений 1 и 2. Выражение 1 задает начальное значение переменной цикла. После каждого исполнения тела цикла значение переменной цикла увеличивается на 1. Цикл завершается, когда значение переменной цикла станет больше значения выражения 2 или после исполнения в теле цикла оператора выхода из цикла (см. далее).

Если оборот FOR отсутствует, а оборот TO присутствует, то исполнение оператора цикла начинается с вычисления выражения 2, целая часть значения которого задает количество исполнений тела цикла. Такой цикл также завершается после исполнения в его теле оператора выхода из цикла.

Если оборот FOR присутствует, а оборот TO отсутствует, то исполнение оператора цикла начинается с вычисления выражения 1, которое задает начальное значение переменной цикла. После каждого исполнения тела цикла переменная цикла увеличивается на 1. Такой цикл завершается только после исполнения в его теле оператора выхода из цикла.

При отсутствии оборотов FOR, TO цикл завершается только после исполнения в его теле оператора выхода из цикла.

⟨оператор выхода из цикла⟩ : : =
'AT' ⟨выражение⟩ 'GO' {⟨оператор⟩} 'OUT' ;

Выражение должно вырабатывать результат типа Logical. Данный оператор должен быть текстуально вложен в оператор цикла. При исполнении оператора выхода из цикла вычисляется выражение, следующее за AT. Если значение выражения истинно, то исполняются операторы между GO и OUT и объемлющий цикл завершается. Если значение выражения ложно, то исполнение оператора выхода из цикла завершается.

⟨оператор задания состава объекта⟩ : : =
['MODEL' ⟨имя модели⟩] 'MADE' 'OF'
[{ ⟨композиция объектов⟩ ';' }]
'FO' ;

Оборот MODEL отсутствует, если данный оператор находится среди предложений в определении объекта. Исполнение такого оператора происходит при вызове содержащего его объекта и состоит в вычислении заданных композиций объектов и связывании результатов вычислений с объектом, содержащим оператор, т. е. вызванному объекту соответствуют результаты исполнения вложенных в него операторов задания состава объекта.

При использовании оператора вне определения объекта оборот MODEL сопоставляет имя заданным композициям объектов. Это имя используется для разрешения ссылок на сгруппированные в операторе объекты в базе данных ССВ. Исполнению оператора задания состава объекта вне определения объекта соответствует вывод результатов вычисления композиций объектов в объектный файл, который соответствует VML-программе.

⟨композиция объектов⟩ : : =
(⟨операция обращения поверхности объекта⟩ ⟨вызов объекта⟩) |
(⟨композиция объектов⟩⟨эйлеровская операция⟩⟨композиция объектов⟩) |
('(' ⟨композиция объектов⟩ ')');

⟨операция обращения поверхности объекта⟩ ⟨объект⟩ : : = 'neg' ;

Эта операция меняет ориентацию граней, задающих поверхность объекта, на противоположную.

⟨эйлеровская операция⟩ : : = + | − | * ;

Поверхности, образуемые гранями первого операнда эйлеровской операции, для краткости далее будут называться поверхности 1, а поверхности, образуемые гранями второго операнда, — поверхности 2. Пусть все поверхности 1 и 2 замкнуты, не самопересекаются и ограничивают ненулевые объемы, называемые далее соответственно объемы 1 и 2. Если взаимное пересечение поверхностей 1 и 2 также ограничивают ненулевые объемы, то поверхности 1 могут частично или полностью находиться внутри объемов 2, а поверхности 2 — внутри объемов 1. Поверхности 1, которые частично находятся внутри объемов 2, обозначаются поверхности 1_внутри_объемов 2, оставшиеся части поверхностей 1 — поверхности 1_вне_объемов 2. Для поверхностей 2 и объемов 1 соответственно используются обозначения поверхности 2_внутри_объемов 1 и поверхности 2_вне_объемов 1. По определению все такие части поверхностей не вырождены и содержат, по крайней мере, по одной грани или по одному «куску» грани соответствующих операндов.

Аддитивная операция + возвращает грани, которые представляют собой эйлеровскую сумму граней операндов и состоят из копий граней поверхностей 1_вне_объемов 2 и поверхностей 2_вне_объемов 1.

Аддитивная операция — возвращает грани, которые представляют собой эйлеровскую разность граней операндов и состоят из копий граней поверхностей 1_вне_объемов 2 и копий граней поверхностей 2_внутри_объемов 1 с обратной ориентацией.

Мультипликативная операция * возвращает грани, которые представляют собой эйлеровское пересечение граней операндов и состоят из копий граней поверхностей 1_внутри_объемов 2 и поверхностей 2_внутри_объемов 1.

⟨вызов объекта⟩ : : =
⟨имя объекта⟩ '(' [⟨фактические параметры⟩] ')';
⟨фактические параметры⟩ : : =
⟨выражение⟩ [{ ','⟨выражение⟩ }];

Фактические параметры при вызове объекта передаются по значению, поэтому в качестве параметров могут использоваться любые выражения, вырабатывающие результаты типов соответствующих формальных параметров в определении объекта.

Выражения. Встроенные переменные и константы, определяемые переменные и формальные параметры объектов имеют имена в отличие от безымянных констант, которые могут быть получены в результате вычисления выражений.

⟨выражение⟩ : : =
⟨операнд⟩ | (⟨выражение⟩⟨двухместная операция⟩⟨выражение⟩)
;
⟨операнд⟩ : : =
('(' ⟨выражение⟩ ')') |
(⟨одноместная операция⟩⟨операнд⟩) |
⟨вызов функции⟩ |
⟨конструктор вектора⟩ |
⟨конструктор последовательности векторов⟩ |
⟨имя константы⟩ |
⟨имя переменной⟩
;

Порядок вычисления выражений определяется приоритетом операций, а также при помощи круглых скобок. Сначала вычисляется часть выражения в скобках. В части выражения, не содержащей скобок, операции с большим приоритетом выполняются раньше операций с меньшим приоритетом, а операции с одинаковыми приоритетами выполняются слева направо. Наибольший приоритет имеют встроенные одноместные операции, далее в порядке убывания приоритета следуют двухместные мультипликативные операции, аддитивные операции, операции сравнения. При вызове функций или объектов сначала вычисляются все фактические параметры в порядке их следования при вызове, затем они ставятся в соответствие формальным параметрам и происходит вызов функции или объекта. Аналогичен порядок вычислений в конструкторах векторов и последовательностей векторов.

Заключение. Предлагаемый язык геометрического моделирования поверхностей трехмерных объектов предоставляет средства конструирования поверхностей вращения, сплайн-поверхностей, выпуклых оболочек и других объектов-примитивов, а также обеспечивает конструирование объектов путем применения эйлеровских (теоретико-множественных) операций: объединения, пересечения, вычитания. VML используется в системе подготовки баз данных [1, 2] для ССВ «Альбатрос» [3], разработанной в ИАиЭ СО РАН. Направлением дальнейшего развития языка является его приближение к алгоритмическим языкам общего назначения, в частности, введение возможностей определения процедур и функций, средств определения новых типов, расширение набора операторов.

СПИСОК ЛИТЕРАТУРЫ

1. **Гусев А. В., Ивашин С. Л., Иоффе А. В., Талыкин Э. А.** Программные компоненты синтезирующих систем визуализации // Автометрия.—1986.—№ 4.
2. **Белаго И. В., Некрасов Ю. Ю., Романовский А. В., Тарасов Ю. В.** Комплекс трехмерного визуального моделирования SoftLab Images 1.1 // Автометрия.—1993.—№ 5.
3. **Долговесов Б. С.** Архитектура систем отображения трехмерных объектов в реальном времени широкого назначения // Машинная графика 89: Программа и тез. докл. V Всесоюз. конф.—Новосибирск: ИАиЭ СО АН СССР, 1989.
4. **Белаго И. В., Романовский А. В., Тарасов Ю. В.** SDL — язык описания визуальных моделей трехмерных сцен // Автометрия.—1993.—№ 5.

Поступила в редакцию 15 июня 1993 г.