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

УДК 681.3.06

И. В. Белаго, А. В. Романовский, Ю. В. Тарасов
(Новосибирск)

SDL — ЯЗЫК ОПИСАНИЯ ВИЗУАЛЬНЫХ МОДЕЛЕЙ ТРЕХМЕРНЫХ СЦЕН

Описывается базовый язык моделирования трехмерных визуальных сцен. Язык универсален как в отношении средств моделирования, так и в отношении используемого типа систем отображения — от программной системы отображения на персональной ЭВМ до высокопроизводительных программно-аппаратных систем реального времени. По этой причине в языке имеется ряд конструкций, избыточных для простых систем отображения. Язык SDL является частью системы SoftLab Images 1.1, продолжившей линию программного обеспечения синтезирующих систем визуализации (ССВ) «Аксай» и «Альбатрос».

Введение. SDL является базовым языком описания визуальных моделей трехмерных сцен в системах синтезирующей визуализации (ССВ) «Аксай» и «Альбатрос» [1—3]. Язык универсален как в отношении моделируемых сцен, так и в отношении к используемому типу систем отображения — от программной системы отображения на персональной ЭВМ до высокопроизводительных программно-аппаратных систем реального времени. В вычислительной модели, реализованной в языке SDL, специфика визуального моделирования по возможности сведена к программированию на традиционных языках высокого уровня. Исполнение SDL-программы соответствует построению одного кадра на устройстве отображения ЭВМ. Последовательности кадров соответствует повторяющееся исполнение программы. Изменение между последовательными кадрами значений, определенных в программе глобальных переменных средствами анимационной системы, передает динамику моделируемой сцены.

Для формального определения языка используется нотация Бэкуса — Наура. В этой нотации терминалы могут выделяться для наглядности двумя парными одиночными или двойными кавычками. Если кавычки являются частью терминала, то это явно оговаривается. Нетерминальные понятия всегда ограничиваются парными угловыми скобками ⟨ ⟩. Метасимвол : : = разделяет определяемое нетерминальное понятие слева от метасимвола и его определение справа от метасимвола. Определение завершается точкой с запятой. Опциональная часть определения нетерминального понятия изображается в квадратных скобках [ ], часть определения, повторяющаяся один или более раз, — в фигурных скобках { }. Соответственно часть определения, повторяющаяся нуль или более раз, изображается в двойных парных скобках [{ }]. Альтернативные части определения, состоящие более чем из одного терминала и не заключенные в квадратные или фигурные скобки, ограничиваются круглыми скобками ( ). Метасимвол | разделяет альтернативные части определения. Пробелы и переносы на новую строку в синтаксических определениях используются для разделения терминалов и метасимволов, а также из эстетических соображений. После формального определения синтаксиса нетерминального понятия следует определение его семантики в свободной нотации и примеры.

Алфавит и словарь языка. Алфавит языка состоит из букв, цифр и прочих графем ASCII, словарь языка (множество лексем) — из служебных лексем,

имен, строковых и числовых литералов, комментариев. Лексемы языка конструируются из графем алфавита языка. Строчные и прописные буквы во всех лексемах, кроме строковых литералов, не различаются. Все лексемы, за исключением строковых литералов и комментариев, не могут содержать пробелов и, кроме скобочных комментариев, не разрываются между строками. При необходимости лексемы отделяются друг от друга пробелами, переносами на новую строку и комментариями. Во всех остальных случаях наличие пробелов, пустых строк, комментариев до и после любой лексемы, запись любой лексемы с новой строки синтаксически безразличны.

```
⟨служебная лексема⟩ : : =
and | animated | begin | by | case | chain | change | check | do |
global | else | external | face | fade | fi | fire | for | from |
if | in | line | model | new | not | od | on | or | out | set | step |
sort then | to | under |'+' | '–' | '*' | ':' | ';' | '(' | ')' |
'=' | '[' | ']' | '{' | '}' | ':=' | '⟨' | '⟨=' | '⟨⟩' | '⟨(' | ')⟩' |
')=' | '/' | ',' | '(*' | '*)' | '⟨*' | '*⟩' | '($' | '$)' ;
⟨имя⟩ : : = ⟨несокращенное имя⟩ | ⟨сокращенное имя⟩ ;
⟨несокращенное имя⟩ : : = ⟨буква⟩ [{⟨буква⟩ | ⟨цифра⟩ | '_'} ] ;
⟨сокращенное имя⟩ : : = '$'⟨буква⟩ [{⟨буква⟩ | ⟨цифра⟩ | '_'} ] ;
```

Имя допускается сокращать справа при условии сохранения однозначности его распознавания. Любое имя может быть сокращено.

Положение '_' в именах является значимым. 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 SDL нет средств для определения типов данных, так как набор встроенных типов данных языка считается достаточным для описания геометрических моделей достаточно представительного класса 3-мерных сцен. Встроенные типы данных представлены логическим типом (Boolean), числовыми типами — целым (Integer) и вещественным (Real), типами векторов в трехмерном пространстве (Vector), заголовков графических примитивов (Header), списков вершин (Vertices), матриц аффинного преобразования (Matrix), систем координат (Reper), списков источников освещения (Lanterns), туманов (Fog), наблюдателей (Observer), сфер (Sphere), плоскостей (Plane).

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 целого типа состоит из целых чисел, которым соответствуют числовые литералы из словаря языка. Множество значений вещественного типа состоит из вещественных чисел, которым соответствуют числовые литералы из словаря языка. К данным числовых типов 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. Если один из операндов двухместной операции принадлежит вещественному типу, а другой операнд — целому, то последний автоматически преобразуется в объект данных вещественного типа.

Множество значений типа векторов в трехмерном пространстве состоит из троек значений числового типа. Первое значение в тройке соответствует координате X вектора, второе — координате Y, третье — 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 умножений, 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 *, /.

Для задания визуальных свойств графических примитивов, которыми в языке SDL являются ломаная линия, грань и множество огней, используются значения типа заголовка графических примитивов. Значения этого типа представляются агрегатами визуальных характеристик, соответствующими каждому графическому примитиву. Встроена операция конструирования данных типа заголовка графических примитивов:

⟨конструктор заголовка графических примитивов⟩ : : =
'{' ⟨ключевой параметр⟩ { ⟨ключевой параметр⟩} '}' ;
⟨ключевой параметр⟩ : : = ⟨сокращаемое имя⟩ '=' ⟨список выражений⟩ ;
⟨список выражений⟩ : : = ⟨выражение⟩ { ','⟨выражение⟩ } ;

Тот или иной набор ключевых параметров определяет, является ли сконструированный объект данных заголовком грани или ее вершины, заголовком множества огней или отдельного огня, заголовком ломаной линии.

Характеристики грани отражают визуальные свойства поверхности моделируемого объекта. Ниже приводится перечень допустимых визуальных характеристик и соответствующих им ключевых параметров, которые можно задавать в заголовке грани.

$color — цвет грани. Задается тремя вещественными выражениями. Значения выражений должны находиться в промежутке [0 . . 1]. Они определяют значения цвета по красной, зеленой и синей компонентам.

$intensity — интенсивность цвета грани. Зависит от наличия и расположения источников освещения, задается вещественным выражением. Значение выражения должно находиться в промежутке [−1 . . 1]. Отрицательное значение интенсивности указывает на то, что грань будет окрашиваться независимо от расположения источников освещения, причем интенсивность цвета определяется ее абсолютным значением.

$transparency — прозрачность грани. Задается двумя вещественными и одним целым выражением. Значения вещественных выражений должны находиться в промежутке [0 . . 1]. Значение 0 соответствует абсолютной непрозрачности, а значение 1 — максимальной прозрачности. Первые два параметра определяют прозрачность в случаях, когда плоскость грани перпендикулярна и параллельна направлению наблюдения соответственно.

$reflection — параметры бликования грани. Блик может возникнуть только у динамической грани, если угол направления свечения источника освещения на нее равен углу направления взгляда наблюдателя. Задаются одним вещественным и одним целым выражением. Первая величина определяет соотношение диффузного и зеркального отражений грани (если она равна t. то из всего отраженного гранью света (1 – t) относится к диффузному отражению, а t — к зеркальному), и ее значение должно находиться в промежутке [0 . . 1 1. Вторая величина определяет степень «полированности» поверхности грани, т. е. ширину блика.

$map — номер текстурной карты. Задается целым выражением. Для каждой вершины такой грани должны быть заданы текстурные координаты.

$dynamic — признак динамической грани. Задается логическим выражением. Если его значение равно true, то цвет грани будет изменяться в динамике в зависимости от взаимного расположения графического примитива и источника освещения.

$shadow — признак расположения грани или линии в тени. Задается логическим выражением. Если его значение равно true, то считается, что соответствующие грани освещаются только рассеянным светом.

$owncolor — признак индивидуального задания цвета. Задается логическим выражением. Для индивидуально заданных цветов с одинаковыми компонентами в памяти отводится одно и то же место.

Пример конструктора заголовка грани:

```
{$col=1,1,1 $ref=1.0,10 $dyn=true}
```

Некоторые визуальные характеристики могут быть заданы индивидуально в каждой вершине грани. При отображении такой грани происходит интерполяция соответствующих индивидуальных характеристик вдоль ее поверхности. В целях имитации гладких неплоских поверхностей в вершинах грани могут быть заданы векторы нормалей, соответствующие градиентам кривизны поверхности в данных точках. Система отображения использует эти данные для интерполяции интенсивности цвета грани вдоль ее поверхности. Любая из визуальных характеристик, заданная в заголовке одной из вершин, должна быть задана и в заголовках всех остальных вершин списка. Ниже приводится перечень допустимых визуальных характеристик и соответствующих им ключевых слов, используемых в заголовках отдельных вершин граней.

$color — цвет грани в данной вершине. Задается так же, как и в заголовке грани.

$intensity — интенсивность цвета в вершине грани. Задается так же. как в заголовке грани.

$transparency — прозрачность в вершине грани. Задается так же, как в заголовке грани.

$normal — нормаль поверхности в вершине. Значение параметра — векторное выражение.

$texture — текстурные координаты вершины. Задаются при помощи двух вещественных выражений. Значения выражений определяют координаты

вершины на плоскости текстурной карты, номер которой задан в заголовке грани.

Далее приводится перечень допустимых визуальных характеристик для огней и соответствующих им ключевых параметров. Любая из перечисленных характеристик может быть задана как в заголовке множества огней, так и индивидуально — в заголовке для отдельного огня.

$color — цвет огней множества. Задается так же, как в заголовке грани.

$diameter — диаметр огней множества. Задается вещественным выражением.

$visibility — дальность видимости огней множества. Задается вещественным выражением.

$normal — направление свечения огней множества. Значение параметра — векторное выражение. Если направление свечения не задано, то считается, что огонь виден с любого направления. В противном случае огонь будет виден только в том полупространстве, куда указывает вектор направления. Дополнительное ограничение видимости может вноситься параметрами $pyramid и $cone.

$pyramid — пирамида видимости огней множества. Задается одним векторным и двумя вещественными выражениями, разделенными запятыми. Векторное выражение (вектор вертикали) вместе с вектором направления определяют плоскости вертикальной и горизонтальной симметрии пирамиды видимости огней множества. Плоскость вертикальной симметрии проходит через указанные векторы, а плоскость горизонтальной симметрии перпендикулярна ей и включает в себя вектор направления. Значения вещественных выражений задают в радианах углы видимости по горизонтали и вертикали соответственно.

$cone — конус видимости огней множества. Задается вещественным выражением, определяющим угол конуса видимости в радианах. Нормаль конуса задается вектором направления свечения.

$blink — номер функции мигания огней множества. Задается целым выражением.

$switch — номер выключателя огней множества. Задается целым выражением.

Визуальным характеристикам линии соответствуют ключевые параметры $color, $intensity, $dynamic, $owncolor с той же семантикой, что и в заголовке грани.

Значения типа списка вершин используются для задания вершин, по которым строятся графические примитивы. Отдельной вершине списка соответствует точка (вектор) в трехмерном пространстве и, возможно, пустой набор визуальных характеристик в данной точке. Набор визуальных характеристик представляется заголовком графического примитива и может косвенно указывать, соответствует ли список вершин грани множеству огней или ломаной линии. Встроена операция конструирования списка вершин.

```
⟨конструктор списка вершин⟩ : : =
'['
[⟨выражение типа Header⟩':' ]⟨описание вершин⟩{','⟨описание вершин⟩}
']' ;
⟨описание вершин⟩ : : = ⟨описание вершины⟩|⟨промежуток⟩ ;
⟨промежуток⟩ : : =
'chain' ⟨выражение типа Integer⟩
'from' ⟨описание вершины⟩ 'to' ⟨описание вершины⟩ ;
⟨описание вершины⟩ : : =
⟨выражение типа Vector⟩ [⟨выражение типа Header⟩] ;
```

К данным типа Vertices применима двухместная аддитивная операция конкатенации, для обозначения которой используется служебная лексема +. В результате исполнения операции производится конкатенация как заголовков, так и перечней описаний вершин операндов. Если в заголовках конкатени-

руемых списков вершин задана одна и та же визуальная характеристика, то выбирается ее значение, определенное в заголовке первого операнда.

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 Vertices и Matrix применима двухместная мультипликативная операция преобразования координат вершин списка, возвращающая результат типа Vertices. Для обозначения этой операции используется служебная лексема *.

Данные типа систем координат предназначены для описания иерархии трехмерных систем координат в моделируемой сцене. Значению этого типа соответствует агрегат, первая компонента которого является указателем на базисную систему координат в описываемой иерархии, а вторая — матрицей перехода из данной системы координат в базисную. В качестве базисной системы координат могут выступать предопределенная мировая система координат, для обозначения которой используется имя world, текущая система координат, установленная оператором задания текущей системы координат (см. ниже), или любая из определенных ранее в SDL-программе систем координат. Встроена операция конструирования системы координат.

⟨конструктор систем координат⟩ : : =
['under' ⟨выражение 1 типа Reper⟩]
'by' ⟨выражение 2 типа Matrix⟩ ;

Выражение 1 задает базисную систему координат, а выражение 2 — матрицу перехода в базисную систему координат. При отсутствии оборота under базисной системой координат считается система координат, установленная оператором задания текущей системы координат.

Данные типа Lanterns используются для задания невидимых источников освещения моделируемой сцены. Встроена операция конструирования списка источников освещения, синтаксис которой аналогичен синтаксису конструктора списка вершин. Координаты вершины в конструкторе задают положение источника освещения, а заголовок — визуальные характеристики источника.

Далее приводится перечень допустимых ключевых параметров и соответствующих им характеристик.

$color — цвет источника освещения. Задается так же, как в заголовке грани.

$visibility — дальность действия. Задается вещественным выражением.

$normal — направления свечения источника. Задается выражением типа Vector.

$cone — телесный угол (в радианах) конуса свечения источника. Задается вещественным выражением.

$reper — система координат источника освещения. Задается выражением типа Reper.

$switch — номер выключателя источника освещения. Задается целым выражением.

Пример конструктора источника освещения:

```
{V1 {$v=1000 $n=-Z $con=P1/6 $col=0,0,1} }
```

Данные типа Fog используются для задания цвета и выборочной непрозрачности среды в моделируемой сцене. Встроена операция конструирования объекта данных типа Fog, синтаксис которой аналогичен синтаксису конструктора заголовка. Далее приводятся ключевые параметры и соответствующие им характеристики.

$visibility — расстояние, на котором прозрачность среды уменьшается в 2 раза. Задается вещественным выражением.

$color — цвет среды. Задается так же, как в заголовке грани.

Объекты данных типа Observer используются при построении изображения моделируемой сцены на экране дисплея компьютера. Встроена операция конструирования объекта данных типа наблюдателей, синтаксис которой аналогичен синтаксису конструктора заголовка. Далее приводятся ключевые параметры и соответствующие им характеристики.

$reper — собственная система координат. Задается выражением типа Reper.

$boundation — пирамида видимости. Задается тремя вещественными выражениями. Значение первого выражения определяет расстояние до плоскости окна наблюдения, значения второго и третьего — размеры окна наблюдения по горизонтали и вертикали в метрах.

Объекты данных типа Plane используются для задания плоскостей в 3-мерном пространстве. Плоскость определяется вектором нормали и точкой. Встроена операция конструирования объектов типа Plane.

```
⟨конструктор плоскости⟩ : : =
'⟨$'⟨ выражение 1 типа Vector⟩','⟨выражение 2 типа Vector⟩ '$⟩' ;
```

Выражение 1 задает нормаль плоскости, выражение 2 — точку плоскости.

К объектам данных типа Plane и Matrix применима двухместная мультипликативная операция * преобразования направления нормали плоскости и координат точек плоскости.

Объекты данных типа Sphere используются для задания сфер в 3-мерном пространстве.

```
⟨конструктор сферы⟩ : : =
'⟨$' ⟨выражение 1 типа Vector⟩','⟨выражение 2 типа Real⟩ '$⟩' ;
```

Выражение 1 задает координаты центра сферы, а выражение 2 — радиус сферы.

К объектам данных типа Sphere и Matrix применима двухместная мультипликативная операция * преобразования координат центра сферы.

Определение и объявление переменных. Переменные определяются до их использования. В определении переменной указывается ее область доступ-

ности, тип данных, имя и, возможно, выражение, в результате исполнения которого переменной присваивается начальное значение.

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

Если определение переменной начинается с лексемы animated, то такая переменная называется анимационной и ее значение может быть изменено анимационной программой.

Если определение переменной начинается с лексемы global, то такая переменная называется глобальной, может быть объявлена внешней и может использоваться в других единицах компиляции, из которых состоит SDL-программа. Анимационные переменные также являются глобальными.

Возможность определения глобальных переменных и объявления их внешними позволяет использовать модульный подход при разработке SDL-программ.

⟨объявление внешних переменных⟩ : : =
'external'
⟨имя типа данных⟩ ⟨имя переменной⟩ {','⟨имя переменной⟩} ';' ;

Определение и объявление моделей. Процедурный механизм в языке SDL реализован в терминах моделей.

⟨определение модели⟩ : : =
['global']
'model' ⟨заголовок модели⟩ '='
'begin' {⟨определение переменной⟩}{⟨оператор⟩} 'end' ';' ;

Модели, как и переменные, могут быть определены как глобальные.

⟨заголовок модели⟩ : : =
⟨имя модели⟩ [⟨формальные параметры модели⟩] ;
⟨формальные параметры модели⟩ : : =
'('
⟨определение формальных параметров⟩
[{ ';'⟨определение формальных параметров⟩ }]
')' ;
⟨определение формальных параметров⟩ : : =
⟨имя типа данных⟩
⟨имя формального параметра⟩[{','⟨имя формального параметра⟩}] ;
⟨описание модели⟩ : : = 'external' ⟨заголовок модели⟩ ';' ;

Глобальная модель, определенная глобальной в одной единице компиляции, может вызываться в другой единице компиляции, если там она объявлена внешней.

Операторы языка.

⟨оператор⟩ : : =
[
⟨оператор присваивания⟩ |
⟨условный оператор⟩ |
⟨оператор цикла⟩ |
⟨вызов модели⟩ |
⟨генератор граней⟩ |
⟨генератор огней⟩ |
⟨генератор ломаных линий⟩ |

⟨оператор задания характеристик среды моделирования⟩|
⟨оператор задания текущей системы координат⟩|
⟨оператор сортировки по плоскостям⟩|
⟨динамический условный оператор⟩|
⟨оператор плавной смены уровня детализации⟩
] ';' ;

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

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

⟨условный оператор⟩ : : =
'if' ⟨выражение типа Boolean⟩
'then' {⟨оператор⟩}
['else' {⟨оператор⟩}]
'fi' ;

Если значение выражения равно true, то исполняются операторы после лексемы then, иначе исполняются операторы после лексемы else, если таковые имеются, и исполнение условного оператора завершается.

⟨оператор цикла⟩ : : =
'for' ⟨имя переменной цикла⟩': ='⟨выражение 1 типа Integer⟩
['step' ⟨выражение 2 типа Integer⟩]
'to' ⟨выражение 3 типа Integer⟩ 'do'
{⟨оператор⟩}
'od' ;

Переменная цикла должна иметь тип Integer. Исполнение оператора цикла начинается с вычисления выражения 1, выражения 2, если таковое есть, и выражения 3. Выражение 1 задает начальное значение переменной цикла. После каждого исполнения тела цикла значение переменной цикла увеличивается на 1 при отсутствии выражения 2 или на значение выражения 2 в противном случае. Цикл завершается, когда значение переменной цикла станет больше значения выражения 3.

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

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

⟨генератор граней⟩ : : =
'face' ⟨выражение типа Header⟩ 'on'
⟨выражение типа Vertices⟩{','⟨выражение типа Vertices⟩} ;

Каждому выражению типа Vertices будет соответствовать грань, геометрические и визуальные характеристики которой определяются значением заголовка и соответствующего списка вершин. Исполнению этого оператора будет соответствовать построение изображения граней при условии их видимости в окне обзора текущего наблюдателя. Считается, что отображаемые грани находятся в текущей системе координат.

⟨генератор огней⟩ : : =
'fire' ⟨выражение типа Header⟩ 'on'
⟨выражение типа Vertices⟩{','⟨выражение типа Vertices⟩} ;

Каждому выражению типа Vertices будет соответствовать множество огней, геометрические и визуальные характеристики которого определяются значением заголовка и соответствующего списка вершин. Исполнению этого оператора будет соответствовать построение изображения огней при условии их видимости в окне обзора текущего наблюдателя. Считается, что отображаемые огни находятся в текущей системе координат.

⟨генератор ломаных линий⟩ : : =
'line' ⟨выражение типа Header⟩ 'on'
⟨выражение типа Vertices⟩{','⟨выражение типа Vertices⟩} ;

Каждому выражению типа Vertices будет соответствовать ломаная линия, геометрические и визуальные характеристики которой определяются значениями заголовка и соответствующего списка вершин. Исполнению этого оператора будет соответствовать построение изображения ломаной при условии ее видимости в окне обзора текущего наблюдателя. Считается, что отображаемая ломаная находится в текущей системе координат.

⟨оператор задания характеристик среды моделирования⟩ : : =
'change' ('modify' | 'substitute') ⟨ключевой параметр⟩
{⟨оператор⟩}
'end' ;

Если после лексемы change присутствует лексема substitute, то новые текущие характеристики среды моделирования становятся равными значению ключевого параметра. Если после лексемы change присутствует лексема modify, текущие характеристики среды моделирования вычисляются как сумма прежнего значения и значения соответствующего ключевого параметра. Текущие характеристики среды моделирования используются при построении изображения (части) моделируемой сцены, получаемого в результате исполнения операторов, вложенных в оператор change.

Ключевые параметры, перечисляемые далее, указывают на изменяемые характеристики среды моделирования и задают значения соответствующих изменений.

$lanterns — список источников освещения для моделируемой сцены (ее части). Задается выражением типа Lanterns.

$fog — туман среды моделирования для сцены (ее части). Задается выражением типа Fog.

$transparency — прозрачность граней моделируемой сцены (ее части). Задается выражением типа Real в интервале [0 . . 1].

$dispersion — коэффициент рассеяния света средой моделирования. Задается выражением типа Real в интервале [0 . . 1].

⟨оператор задания текущей системы координат⟩ : : =
'set' ⟨выражение типа Reper⟩
{⟨оператор⟩}
'end'

Задаваемая выражением система координат становится текущей на время исполнения вложенных операторов. После исполнения данного оператора восстанавливается прежняя текущая система координат. Считается, что все выражения типа Vector, Vertices, Plane, Sphere, используемые во вложенных операторах, задают значения относительно текущей системы координат.

⟨оператор сортировки по плоскостям⟩ : : =

'sort' ['by'] ⟨выражение типа Plane⟩
{⟨оператор⟩}
'::'
{⟨оператор⟩}
'end'

Если наблюдатель расположен выше плоскости, задаваемой выражением, то сначала исполняется первая вложенная последовательность операторов, а затем — вторая. В противном случае порядок исполнения обратный.

⟨динамический условный оператор⟩ : : =
'check' ⟨имя динамической функции⟩ ⟨фактические параметры⟩
{⟨оператор⟩}
[
'::'
{⟨оператор⟩}
]
'end'
⟨имя динамической функции⟩ : : =
'VISUSPH' | 'POSUSPH' | 'POSISPH' | 'POSUPLN' | 'POSIPLN' | 'OBSLIST';

При исполнении оператора вызывается динамическая функции, соответствующая указанному имени с заданными фактическими параметрами. Если функция возвращает true, то исполняется первая вложенная последовательность операторов и исполнение динамического условного оператора завершается. Если функция возвращает false, то исполняется вторая вложенная последовательность операторов, если таковая имеется, и исполнение динамического условного оператора также завершается.

Функция с именем VISUSPH может иметь один или более параметров типа Sphere. Функция возвращает true, если в пирамиду видимости текущего наблюдателя попадает частично или полностью хотя бы одна из сфер-параметров. Если ни одна из сфер не попадает в пирамиду видимости текущего наблюдателя, то функция возвращает false.

Функция с именем VISISPH может иметь один или более параметров типа Sphere. Функция возвращает true, если текущий наблюдатель находится внутри хотя бы одной из указанных сфер, иначе функция возвращает false.

Функция с именем POSUPLN может иметь один или более параметров типа Plane. Функция возвращает true, если текущий наблюдатель находится выше хотя бы одной из указанных плоскостей, иначе функция возвращает false.

Функция с именем POSIPLN может иметь один или более параметров типа Plane. Функция возвращает true, если текущий наблюдатель находится выше всех указанных плоскостей, иначе функция возвращает false.

Функция с именем OBSLIST может иметь один или более параметров типа Observer. Функция возвращает true, если текущий наблюдатель совпадает хотя бы с одним из указанных параметров-наблюдателей, иначе функция возвращает false.

⟨оператор выбора⟩ : : =
'case' ⟨выражение типа Integer⟩
{ '::' ⟨целое число 1⟩ | '<< ' ⟨целое число 2⟩] {⟨оператор⟩} }
['::' {⟨оператор⟩}]
'end'

Целое число 1 и целое число 2 называются метками варианта. Одно ⟨целое число 1⟩ задает интервал из одной метки, а пара целых чисел — интервал из нескольких меток. Интервалы меток не должны перекрываться. Исполнение оператора выбора начинается с вычисления выражения типа Integer. Если результат выражения попадает в интервал меток варианта, то выполняется

соответствующая последовательность операторов и исполнение оператора выбора завершается. Если результат выражения не попадает ни в один из интервалов меток, то исполняются операторы варианта без меток после лексемы '::' при их наличии и исполнение оператора выбора также завершается.

⟨оператор плавной смены уровня детализации⟩ : : =
'fade'
['in']⟨выражение 1 типа Real⟩[','⟨выражение 2 типа Real⟩]
['out']⟨выражение 3 типа Real⟩[','⟨выражение 4 типа Real⟩]
['by'] ⟨выражение 5 типа Vector⟩
{⟨оператор⟩}
'end'

Если выражение 2 отсутствует, то вместо его результата используется результат выражения 1. Если выражение 4 отсутствует, то вместо его результата используется результат выражения 3. Значения выражений 1 и 2 задают интервал проявления части сцены, изображение которой строится в результате исполнения вложенных в данный оператор операторов, а значения выражений 3 и 4 задают интервал исчезновения этой части сцены. Выражение 5 задает визуальный центр части сцены, до которого рассчитывается расстояние до текущего наблюдателя. Если рассчитанное расстояние меньше значения выражения 1 или больше значения выражения 4, то вложенные операторы не исполняются. Если расстояние находится в интервале значений выражений 2 и 3, то изображение сцены строится полностью проявленным.

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 сопоставляются с формальными параметрами и происходит вызов функции или объекта. Аналогичен порядок вычислений в конструкторах векторов и последовательностей векторов.

Макросредства. В препроцессор языка SDL встроен универсальный макрогенератор, обеспечивающий произвольную параметризованную текстовую подстановку. К средствам макрогенератора относятся макроопределение и макровызов. Макроопределение позволяет описать подставляемый текст, присвоить ему имя и, возможно, параметризовать его.

⟨макроопределение⟩ : : =
'macro' ⟨имя определяемого макроса⟩
['('
⟨имя формального параметра макроса⟩
[{',' ⟨имя формального параметра макроса⟩}]
')']'=' '⟨*' ⟨тело макроса⟩ '*⟩' ';' ;

Тело макроса может содержать любые лексемы языка, за исключением "⟨*", "*⟩". Макроопределение может располагаться в любом месте программы, но оно должно текстуально предшествовать соответствующему макровызову.

⟨макровызов⟩ : : =
⟨имя вызываемого макроса⟩
['('
⟨фактический параметра макроса⟩
[{ ',' ⟨фактический параметра макроса⟩}]
')'];

При обработке макровызова вместо макроса подставляется его тело, при этом каждое вхождение формального параметра заменяется соответствующим фактическим. Если после такой подстановки при исполнении полученного текста обнаруживаются макровызовы, то описанный процесс повторяется для каждого макровызова. Допускается рекурсивный вызов макросов.

Заключение. Обсуждаемый язык является последней версией языка SDL, описание которого было изложено в [4]. Так как язык в основном ориентирован на моделирование трехмерных визуальных сцен в ССВ «Аксай» и «Альбатрос», то это обусловило наличие в нем ряда конструкций, избыточных для простых систем отображения, и ряда ограничений, связанных с особенностями архитектуры систем реального времени, в частности, отсутствуют средства определения новых типов, функций. Перспективы развития языка связаны с развитием архитектуры ССВ в сторону большей универсализации.

СПИСОК ЛИТЕРАТУРЫ

1. **Гусев А. В., Ивашин С. Л., Иоффе А. В., Талныкин Э. А.** Программные компоненты синтезирующих систем визуализации // Автометрия.—1986.—№ 4.
2. **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.
3. **Белаго И. В., Некрасов Ю. Ю., Романовский А. В., Тарасов Ю. В.** Комплекс трехмерного визуального моделирования SoftLab Images 1.1 // Автометрия.—1993.—№ 5.
4. **Гусев А. В., Талныкин Э. А.** SDL — язык описания трехмерных сцен в системах динамической машинной графики // Автометрия.—1986.—№ 4.

Поступила в редакцию 4 июня 1993 г.