Композиционный подход к построению программных приложений визуализации*

В.А. Семенов, Е.В. Алексеева, С.В. Морозов, О.А. Тарлапан

Аннотация. В статье предлагается подход к построению программных приложений визуализации с использованием формально специфицируемых правил преобразования информационных моделей друг в друга. Визуализация рассматривается как процесс отображения прикладной модели на ее визуальные представления. Подход позволяет формально связать прикладную и визуальную модели, специфицировать сценарий визуализации в виде композиции отображений и затем генерировать требуемые визуальные сцены для конкретных наборов прикладных данных путем выполнения сценария. Будучи реализованным, подход позволяет визуализировать сложные междисциплинарные данные с применением разнообразных, настраиваемых в ходе работы приложения методик в строгом соответствии со специфицируемыми правилами. В результате программные приложения выигрывают в выразительности, а также приобретают дополнительную надежность, гибкость и возможность многократного использования в новых прикладных контекстах.

Обсуждаются различные возможности для эффективной реализации подхода. Они связаны с использованием предложенного декларативного языка потоков данных EXPRESS-F и описываемой инструментальной среды разработки приложений визуализации общего назначения.

Применение подхода иллюстрируется на примере приложения, предназначенного для макетирования архитектурно-строительных проектов с использованием технологий виртуальной реальности.

Ключевые слова: STEP, EXPRESS, IFC, VRML97, X3D.

1. Что такое визуализация? Введение

Что такое визуализация? Ответ на этот вопрос довольно нетривиален. Имеется множество частных примеров «правильной» визуализации, но до сих пор нет методического понимания, что же из того, что составляет хорошую визуализацию, являлось бы привлекательным для естественных моделей, человеческого восприятия и внутреннего зрения. Множество людей определяли понятие визуализации различными способами [1].

* Работа поддержана РФФИ, гранты 01-01-00239, 03-01-06422

Рене Декарт (1637): «Воображение или визуализация, и, в частности, использование диаграмм, играет значительную роль в научных исследованиях».

Александр фон Гумбольд (1811): «Все, что относится к мере и количеству, может быть представлено геометрическими образами. Статистические проекты, которые обращаются к органам чувств, не утомляя разум, обладают преимуществом привлечения внимания к большему числу важных фактов».

Маккормик и др. (1987): «Визуализация — это вычислительный метод. Она преобразует символические образы в геометрические, позволяя исследователям наблюдать их моделирование и вычисление. Визуализация предлагает метод для наблюдения невидимого. Она обогащает процесс научного открытия и способствует глубоким и непредвиденным проникновениям в суть вещей. … Много лет тому назад Ричард Хамминг заметил, что: «цель вычислений — это понимание, а не числа». Цель визуализации заключается в усилении существующих научных методов путем обеспечения нового научного понимания через визуальные методы».

Роговиц (1993): «Визуализация — это процесс отображения численных значений в воспринимаемое познаваемое пространство».

Кроме исторических проблем эволюции знаний, причина неоднозначности взглядов заключается в разнообразии исследуемых проблематик, существенных различиях моделей, подлежащих визуализации, и широком наборе методов и техник, которые бы потенциально подходили для достижения подобных целей. Тем не менее, интересно заметить, что все эти определения сводятся к трансформированию или отображению данных в некоторые воспринимаемые представления, допускающие четкую зрительную интерпретацию. Следовательно, отображение данных в зрительно связанные образы может рассматриваться как основной принцип визуализации в целом и ее многочисленных приложений в частности.

Действительно, традиционный конвейер визуализации может быть представлен как композиция отображений, соответствующих общеизвестным стадиям подготовки исходных прикладных данных, их визуализации, моделирования производных визуальных сцен, в частности, с использованием технологии виртуальной реальности, и, наконец, рендеринга и отображения сцен [2]. Здесь под прикладными данными понимается произвольная структурированная типи-зированная информация, используемая в различных областях промыш-ленности, бизнеса и науки. На первой стадии конвейера исходные данные проходят предварительную обработку для исключения возможной избы-точности, а также для их улучшения и обогащения. При этом не имеет значения, выполняются ли эти отображения над данными, принадлежащими той же самой информационной модели, или разным. Стадия визуализации соответствует отображению данных из одной символической модели в другую, имеющую явное визуальное содержание, такое как пространство, время, форма, звук, и т.д. На следующей стадии полученное

визуальное представление может быть дополнительно преобразовано в сцены моделирования виртуальной реальности, непосредственно поддерживаемые популярными Интернет-браузерами и интерфейсами (API) графических программных систем. На заключительной стадии происходит формирование и отображение изображений моделируемых сцен, что соответствует отображению визуальной сцены в генерируемые цифровые изображения, которые и являются конечной целью процесса визуализации.

Таким образом, все стадии могут рассматриваться как отображения одних данных в другие. Под более пристальным взглядом каждая из этих стадий может быть также представлена как композиция отображений, выполняющих частичные преобразования типизированной информации. Каждое отображение реализует некоторое правило, согласно которому типизированные информационные элементы, принадлежащие одной модели, должны быть трансформированы в соответствующие элементы другой модели. Из-за высокой степени интеграции и сложности разработанных в последнее время многопрофильных информационных моделей, содержащих тысячи взаимосвязанных типов данных, программирование приложений визуализации оказывается трудной задачей, поскольку требует обработки значительной части подобной всеобъемлющей информации. Чтобы визуализировать прикладные данные эффективно и адекватно выбранным аспектам изучаемой проблемы, методы и сценарии визуализации должны опираться на подходящую визуальную метафору. Очевидно, что такие программы могут быть очень сложны и нуждаются в более строгой структуризации, а также в предусмотренных средствах адаптации и расширения с использованием некоторого общего формализма, содержательного для приложений визуализации.

Другая мотивация выработки формализма и следования ему при разработке программных приложений связана с требованием согласованного управления прикладными данными и их визуальными представлениями, существенным для интерактивных систем. В таких приложениях изменения визуальных сцен, вызванные реакциями пользователя, должны передаваться через все стадии конвейера визуализации в обратном порядке, чтобы привести прикладные данные в соответствие с их визуальным образом. Для достижения такой цели могут применяться композиции обратных преобразований.

Например, при проектировании здания архитектор манипулирует визуальными примитивами, такими как ломаные, поверхности, твердые тела, цветовые палитры, непосредственно отображаемыми в графическом контексте приложения. Но эти манипуляции интерпретируются и обрабатываются как операции над прикладными данными, представляющими стены, окна, перекрытия, крыши и так далее, значимыми в этой прикладной области. Подобным образом, при изучении и исследовании магнитных, электрических, гравитационных полей посредством манипулирования визуальными сценами, геофизик фактически восстанавливает прикладные данные, соответствующие значимым в геологии свойствам горизонтов, скважин, областей и т.д.

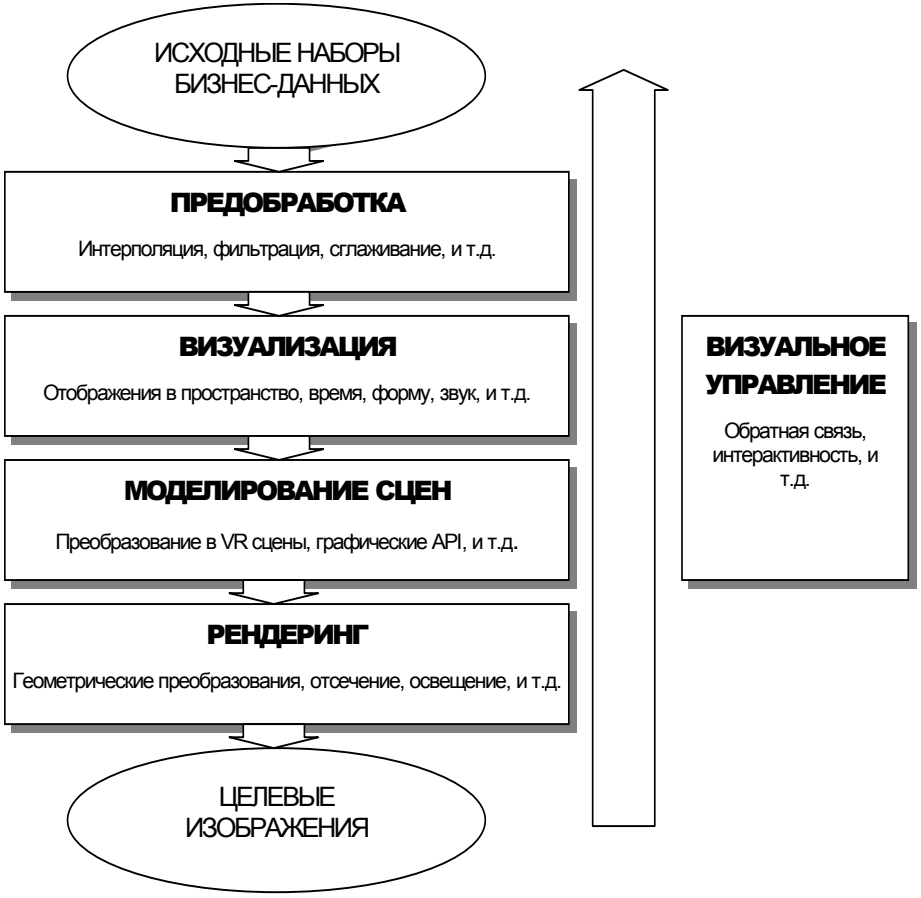

Рис. 1. Представление традиционного конвейера визуализации как композиции отображений

Упомянутые выше обстоятельства побудили нас взглянуть на дисциплину визуализации через призму парадигмы композиции отображений, для того чтобы выработать формальный подход к программированию приложений визуализации и предложить конструктивные способы его реализации.

В разделе 2 описан формальный подход к определению синтаксиса и семантики отображений. При этом мы следуем объектно-ориентированному моделированию информации, относящейся к прикладной и визуальной предметным областям. В разделе 3 представлены декларативный язык EXPRESS-F и инструментальная среда разработки приложений визуализации общего назначения, реализующие формализованный подход. В разделе 4

приводятся некоторые иллюстрирующие его примеры приложений визуализации. Особое внимание в данных примерах уделяется практическим преимуществам предлагаемого подхода, которые кратко резюмируются в заключительном разделе.

2. Формальный подход

Представим подход к визуализации, рассмотрев сначала понятия, представляющие информационную метамодель, подходящую для специфицирования прикладной и визуальной предметных областей. Язык моделирования данных EXPRESS используется здесь с тем, чтобы проиллюстрировать вводимую концепцию и подтвердить ее применимость по отношению к другим языкам и реализационным платформам. Затем приступим к формализации отображений, устанавливающих взаимосвязь между визуальной и прикладной моделями, а также налагающих правила, согласно которым визуальные образы могут быть однозначно соотнесены с исходными прикладными данными. Формализация подхода следует методу, используемому в семантическом анализе языков [3, 4].

Для рассматриваемого подхода не имеет принципиального значения, какой именно язык используется в качестве средства его реализации. В частности, как декларативные, так и исполняемые объектно-ориентированные языки, например, UML, BON, OPEN, C++, Java, Eiffel, C#, потенциально подходят для спецификации и реализации приложений визуализации с использованием представленного ниже подхода.

2.1. Объектно-ориентированная метамодель

Итак, определим объектно-ориентированную метамодель как структуру $M = (C, \pi, Attr_C, Rule_C)$ со следующим значением:

- C — множество классификаторов, которые представляют различные типы объектов. Иерархия подтипов, порождаемая отношениями обобщения/специализации классификаторов, используется в качестве основы для встраиваемого механизма полиморфизма;
- π — частичный порядок на C, отражающий иерархию типов объектов и специализацию классификаторов;
- $Attr_C$ — множество атрибутов или, точнее, множество сигнатур операций $a_c : c \to s$, $a_c : c \times s \to c$ для функций доступа к значениям атрибутов. Каждый атрибут объекта типа $s \in C$ может представлять свойство ассоциации с объектами того же типа $c \in C$;
- $Rule_C$ — множество ограничений, на основе которых определяются поведенческие характеристики объектов. Они могут определять количество, тип и организацию атрибутов, а также налагать ограничения на их значения. Формально, данные правила являются множеством сигнатур операций $r_c : c \to boolean$.

Предполагается, что атрибуты бывают основных типов S_D, отображаемых в основную область семантики D, сложных типов $S_{\overline{D}}$, отображаемых в многозначную область семантики $\overline{D}$, а также типов классов C. Способом, аналогичным алгебраической спецификации, предоставим сигнатуру $\Sigma_M = (S_D \cup S_{\overline{D}} \cup C, \pi, \Omega_D \cup \Omega_{\overline{D}} \cup Attr_C \cup Rule_C)$, где Ω_D, $\Omega_{\overline{D}}$ являются множествами операций, определенных над основными и сложными типами. Здесь частичный порядок π расширяет порядок классификаторов и обобщает отношения специализации между всеми типами данных. Сигнатура, сформированная подобным способом, описывает типы и символы операций, принадле-жащие к информационной модели M, а также содержит исходное множество синтаксических элементов, над которыми могут быть определены выражения $Expr(\{var_s\}_{s \in S_D \cup S_{\overline{D}} \cup C})$ с переменными, индексируемыми данными типами.

Множество объектов со значениями атрибутов и установленными между ними ассоциациями определяет состояние модели σ_M. В рамках нашего обсуждения, состояние σ_M является множеством прикладных или визуальных объектов в зависимости от семантики модели предметной области M.

Для краткости пропустим более глубокую формализацию и перейдем к рассмотрению синтаксиса и семантики языка информационного моделирования EXPRESS [5], чтобы проиллюстрировать введенную метамодель. Отметим лишь, что в настоящее время язык EXPRESS используется в качестве стандартного средства спецификации моделей данных для различных отраслей науки и промышленности.

2.2. Язык моделирования EXPRESS

Итак, EXPRESS определяет основные типы $\{Real, Integer, Number, Boolean, Logical, String, Binary\} \subseteq S_D$, соответствующие простым типам данных с общепринятой семантикой. Сложные типы $\{Generic\ Aggregate, Bag, Set, List, Array, Enumeration, Select, Defined\} \subseteq S_{\overline{D}}$ соответствуют различным типам множеств, перечисляемым типам, выборкам и переопределенным типам, которые допускают определение производных и вложенных структур данных. Возможная дополнительная специализация типов не препятствует дальнейшему обсуждению.

На Рис. 2 представлена общая классификация имеющихся в языке EXPRESS типов. Типы, отмеченные жирным курсивом, соответствуют его исходным типам. Стрелки показывают отношения наследования между ними. Виртуальные типы, отмеченные обычным шрифтом, используются только для того, чтобы единообразно классифицировать конструкции языка и представить типы, определяемые пользователем.

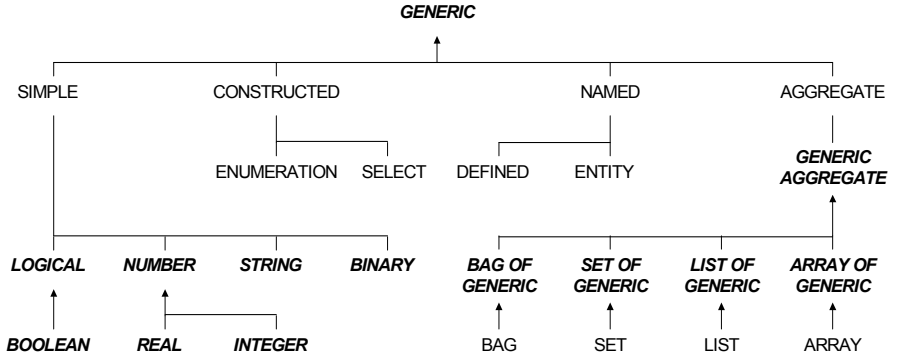

Рис. 2. Классификация исходных типов, имеющихся в языке EXPRESS

Простыми типами данных являются `Boolean`, `Logical`, `Number` (включая конкретные подтипы `Real` и `Integer`), `String` и `Binary`. Синтаксис простых типов и их свойства описываются сигнатурой $\Sigma_D = (S_D, \Omega_D)$, где S_D — множество типов данных, а Ω_D — множество операций. Интерпретация простых типов является обычной, но каждое множество расширяется специальной величиной $\perp$, обозначающей неопределенное значение. Тип $s \in S_D$ отображается в область семантики D_s функцией $I : s \alpha D_s$ следующим образом: `I (Boolean) = {true, false}` Y{⊥}, `I (Logical)` = `{true, false, unknown}` Y{⊥}, `I (Real)` = R Y{⊥}, `I (Integer)` = `Z` Y{⊥}, `I (String)` = A^* Y{⊥}, где A — конечный алфавит, A^* — множество всех последовательностей над алфавитом A, и *I (Binary)* = $\{0,1\}^*$ Y{⊥}.

Над простыми типами определено множество операций. Существует несколько групп операторов:

{*+, -, *, /, **: Number* × *Number* → *Number; +, -, abs: Number* → *Number* } Y

{*div, mod: Integer* → *Integer; odd: Integer* → *Logical* } Y

{*sin, cos, tan, asin, acos, atan, exp, log, log2, log10, sqrt: Number* → *Real* } Y

{*format: Number* × *String* → *String; value: String* → *Number* } Y

{*and, or, xor: Logical* × *Logical* → *Logical; not: Logical* → *Logical* } Y

{*+: String* × *String* → *String, like: String* × *String* → *Logical; length: String* → *Integer; []: String* × *Integer* → *A, [:]: String* × *Integer* × *Integer* → *String*} Y

{*+: Binary* × *Binary* → *Binary; blength: Binary* → *Integer; []: Binary* × *Integer* → *0|1, [:]: Binary* × *Integer* × *Integer* → *Binary* } Y

{*=, <>, <=, >=, <, >: Number* × *Number* → *Logical, Logical* × *Logical* → *Logical, String* × *String* → *Logical, Binary* × *Binary* → *Logical*}.

Семантику данных операций легко истолковать. В частности, это арифметические операции над числовыми операндами типов *Number*, *Real*, *Integer* и стандартные математические функции; преобразование чисел в строковое представление и обратно; логические операции над логическими операндами типов *Logical* или *Boolean*; конкатенация, сопоставление с образцом, индексация, выделение подмножества над операндами типов *String* и *Binary*, а также операции сравнения, определенные для всех простых типов. Некоторые операции имеют тот же самый перегруженный символ в качестве имени и могут различаться только по типам своих аргументов.

Созданные типы, которыми могут являться перечисление либо выборка, расширяют множество основных типов. Область определения перечисляемого типа *Enumeration* задается упорядоченным множеством значений, представленных уникальными именами $I(Enumeration(a_i, i = 1...n)) = \{a_i \in A^*, i = 1...n\}$ Y{⊥}. На литералы перечисляемого типа ссылаются как на элементы перечисления. Порядок элементов перечисления предопределяется их относительными позициями. Данное упорядочение дает в результате операции сравнения, определенные между значениями, относящимися к одному и тому же перечисляемому типу:

{*=, <>, <=, >=, <, >: Enumeration* × *Enumeration* → *Logical* }.

Выбираемый тип данных *Select* определяет производный тип, представляемый списком других исходных типов. Экземпляр выбираемого типа данных является экземпляром одного из типов, заданных в списке выбора. Допускается, что экземпляр данных принадлежит одному из нескольких возможных типов. Другими словами, область определения значений для подобного типа является объединением областей определения типов в его списке: $I(Select(s_i, i = 1...n)) = I(s_1)$ Y ... Y $I(s_n)$ Y{⊥}.

Многозначные выражения в EXPRESS описываются агрегатными типами. Агрегации используются для представления упорядоченных и неупорядоченных коллекций данных каких-либо основных, предопределенных сложных и объектных типов. Они конструируются путем указания типа для их элементов. Эти коллекции могут иметь фиксированные или переменные размеры, хранить уникальные или дублированные представления элементов, а также могут допускать неопределенные элементы, что дает в результате разреженные структуры данных.

Так, тип данных *Array* (массив) представляет собой структуру с фиксированным размером, где существенной является индексация элементов. Опционально массивы могут допускать, что не все их элементы имеют значение. Свойство уникальности массивов означает, что каждый элемент отличается от любого другого в одном и том же экземпляре массива. Тип данных *List* (список) представляет упорядоченную коллекцию идентичных элементов.

Список может хранить любое количество элементов, допуская или, опционально, не допуская их дублирование. Тип данных *Bag* (мультимножество) является коллекцией элементов, в которой порядок не является важным и допускается дублирование. И, наконец, *Set* (набор) является коллекцией элементов, в которой порядок не является важным и дублирование элементов не допускается. Количество элементов в списках, мультимножествах, наборах может изменяться в зависимости от определения их границ. Важно, что агрегатные типы могут быть вложенными, таким образом давая в результате многомерные структуры данных.

{ +, -, *: $Aggregate(s_1) \times Aggregate(s_2) \to Aggregate(s)$ } Y

{ =, <>, <=, >=: $Aggregate(s_1) \times Aggregate(s_2) \to Logical$ } Y

{ *in*: $s \times Aggregate(s) \to Logical$, *query*: $Aggregate(s) \times Logical\ Expression \to Aggregate(s)$ } Y

{ *[]*: $Aggregate(s) \times Integer \to s$ } Y

{ *sizeof, hiindex, loindex, hibound, lobound*: $Aggregate(s) \to Integer$ } Y

{ *insert*: $List(s) \times s \times Integer \to List(s)$, *remove*: $List(s) \times Integer \to List(s)$}.

Операторы над множествами +, -, * определяются для объединения, вычитания и пересечения коллекций совместимых типов s_1, s_2. Операторы сравнения, включая операторы подмножества и надмножества <=, >=, предназначены для установления отношений равенства и общности между совместимыми коллекциями экземпляров. Оператор принадлежности *in* проверяет некоторое значение на наличие в агрегации. Выражение запроса *query* вычисляет логическое условие отдельно для каждого элемента агрегации и возвращает агрегацию, содержащую те элементы, для которых логическое выражение истинно. Оператор индексирования *[]* извлекает из агрегации отдельный элемент. Операции *sizeof*, *hiindex*, *loindex, hibound, lobound* возвращают, соответственно, количество элементов в агрегации, верхний и нижний индексы элементов массива, верхнюю и нижнюю границы списков, мультимножеств и наборов. Операции *insert* и *remove* определяются дополнительно, чтобы упростить манипулирование специфическими агрегациями — списками.

Наконец, определенный (*Defined*) тип данных является пользовательским расширением стандартных типов, при этом он дает возможность определить дополнительные семантические ограничения, налагаемые на специфицируемые данные: $I(Defined(s)) = I(s)\ Y\ \{\perp\}$.

Множества, выборки и определенные типы могут быть вложенными, поэтому в EXPRESS допустимо составление более сложных типов. В нетривиальных прикладных и визуальных моделях данных невозможно избежать подобных сложных структур.

2.3. Определение семантики отображений

При условии, что заданы информационные модели прикладной и визуальной предметных областей, схема отображения может быть определена как система $R_{M_S M_T} = (M_S, M_T, Map, \pi, Pr_{Map}, Ps_{Map}, Link_{Map}, Attr_{Map}, Qr_{Map}, Qs_{Map})$, где:

M_S, M_T — модели исходной и целевой предметных областей. Предполагается, что они могут различаться, пересекаться или совпадать. Если модели совпадают, то схема $R_{M_S M_T}$ соответствует некоторому преобразованию модели M_S саму в себя;

Map — множество двунаправленных типов отображений $m(l'_1,...,l'_k,l''_1,...,l''_l,l'''_1,...,l'''_n) \in Map$ с входными, смешанными и выходными связями $l'_1,...,l'_k,l''_1,...,l''_l,l'''_1,...,l'''_n \in Link_{Map}$, принадлежащими соответствующим типам исходной и/или целевой моделей $s'_1,...,s'_k,s''_1,...,s''_l,s'''_1,...,s'''_n \in S_{D_S}\ Y\ S_{\overline{D}_S}\ Y\ C_S\ Y\ S_{D_T}\ Y\ S_{\overline{D}_T}\ Y\ C_T$. Каждое отображение `m` представляется парой сигнатур $m_f : m \times s'_1 \times ... \times s'_k \times s''_1 \times ... \times s''_l\ \alpha\ s''_1,...,s''_l,s'''_1,...,s'''_n$ для прямой функции, отображающей исходные объекты в целевые, и $m_b : m \times s''_1 \times ... \times s''_l \times s'''_1 \times ... \times s'''_n\ \alpha\ s'_1,...,s'_k,s''_1,...,s''_l$ для обратной функции. Связи играют роль формальных параметров как для функций отображений, так и для других операций, определяемых для концепта отображения. Каждое отображение определяет, какие комбинации типов параметров, включая основные, сложные и объектные типы, могут привести к генерации результатов или восстановлению исходных данных, а также как это может быть выполнено на основе предусловий Pr_{Map}, постусловий Ps_{Map} и операций над атрибутами $Attr_{Map}$. Здесь предполагается, что целевая сцена всегда может быть получена из всеобъемлющей исходной информации, а исходные данные могут реконструироваться при изменениях целевой сцены;

π — частичный порядок на `Map`, отражающий отношения обобщения/специализации между отображениями;

Pr_{Map}, Ps_{Map} — множества предусловий $pr_m : m \times s'_1 \times ... \times s'_k \times s''_1 \times ... \times s''_l\ \alpha\ boolean$ и постусловий $ps_m : m \times s''_1 \times ... \times s''_l \times s'''_1 \times ... \times s'''_n\ \alpha\ boolean$ для выявления потенциальных случаев, когда прямое и обратное преобразования должны действительно выполняться. Они также могут применяться для контроля правильности и непротиворечивости результатов;

$Attr_{Map}$ — множество операций над атрибутами $a_m : m\ \alpha\ s$, $a_m : m \times s\ \alpha\ m$ для функций доступа к их значениям. Каждый атрибут отображения типа

$s \in Map$ представляет свойство ассоциации, задаваемое значением типа $m \in Map$. Таким образом, отображения могут связываться друг с другом;

Qr_{Map}, Qs_{Map} — множества качественных правил $qr_m : m \times s'_1 \times ... \times s'_k \times s''_1 \times ... \times s'' \alpha \ number$, $qs_m : m \times s''_1 \times ... \times s''_l \times s'''_1 \times ... \times s''' \alpha \ number$ для априорной и апостериорной оценки результатов, получаемых при выполнении отображений. Здесь полагается, что степень качества и пригодности результатов может быть выражена типом, для которого определены операции сравнения.

Важно, что порядок типов исходной и целевой моделей индуцирует частичный порядок на отображениях. Отображения $m(l'_1,...,l'_k,l''_1,...,l''_l,l'''_1,...,l'''_n) \in Map$, $\overline{m}(\overline{l}'_1,...,\overline{l}'_{\overline{k}},\overline{l}''_1,...,\overline{l}''_{\overline{l}},\overline{l}'''_1,...,\overline{l}'''_{\overline{n}}) \in Map$ могут быть связаны соотношением $\overline{m} \ \pi \ m$, if $k \le \overline{k}$, $l \le \overline{l}$, $n \le \overline{n}$, $\overline{s}'_i \ \pi \ s'_i$ для $\forall i = 1,...,k$, $\overline{s}''_i \ \pi \ s''_i$, для $\forall i = 1,...,l$, и $\overline{s}'''_i \ \pi \ s'''_i$ для $\forall i = 1,...,n$. Следствием частичного порядка на Map являются условия на идентификаторы данных, которые могут участвовать в отображениях. Если идентификаторы данных удовлетворяют отображению $\overline{m}$ и $\overline{m} \ \pi \ m$, то они удовлетворяют и отображению m. Это означает, что тот же самый набор данных или популяция объектов могут обрабатываться при вычислении родственных отображений. Естественный отбор приводит к применению того отображения, которое наиболее подходит к конкретному набору данных по сигнатуре функции отображения. С этой целью типы данных и связей отображений могут сравниваться методом, схожим с лексикографическим сравнением строк, с тем, чтобы произвести качественную оценку и принять окончательное решение, основанное на вычисляемой мере. Конечно, могут быть другие зависимые от приложения способы определения функций оценки качества для отображений.

Схема отображения $R_{M_S M_T}$ допускает содержательную реализацию $R_{M_S M_T} : \sigma_{M_S} \to \sigma_{M_T}$, преобразующую состояние модели M_S в состояние модели M_T или, иначе говоря, преобразующую популяции объектов, относящихся к модели M_S, в популяции объектов, принадлежащих модели M_T. Если модель M_S семантически означает некоторую прикладную модель, а M_T — некоторую визуальную модель, то данная реализация будет соответствовать процессу визуализации. Более того, будучи схемой двунаправленного отображения, $R_{M_S M_T}$ может допускать реализацию $R_{M_S M_T} : \sigma_{M_T} \to \sigma_{M_S}$, воспроизводящую исходные прикладные данные в соответствии с изменяющейся визуальной сценой. Иногда приложения визуализации предусматривают интерактивные возможности, основанные на манипуляциях объектами визуальной сцены и их преобразованиях в исходное

представление, однако в общем случае не требуется обязательного существования обратных отображений. В последующем обсуждении рассмотрим данный подход только на примерах прямых отображений, допуская, что подобным образом могут применяться и обратные преобразования.

Рассмотрим основные принципы и механизмы, существенные для введенного понятия отображения. Они имеют значение для построения широкого класса приложений с использованием композиционного подхода к программированию. Обсудим данные принципы, следуя приведенному ниже примеру.

```
SCHEMA ConstructiveSolidGeometry

  ENTITY Solid ABSTRACT SUPERTYPE OF (ONEOF(Sphere, Cube,
Cylinder, Cone, Torus, Polyhedron, Box, Half_Space));
    DERIVE
      is_bounded: BOOLEAN := ?;
      bounding_box: Box := ?;
      area_of: REAL := ?;
      volume_of: REAL := ?;
      boundary_representation: Polyhedron := ?;
  END_ENTITY;

  ENTITY Box
    SUBTYPE OF (Solid);
      Xmin: REAL;
      Xmax: REAL;
      Ymin: REAL;
      Ymax: REAL;
      Zmin: REAL;
      Zmax: REAL;
    DERIVE
      SELF\Solid.is_bounded: BOOLEAN := TRUE;
      SELF\Solid.bounding_box: Box := SELF;
      SELF\Solid.area_of: REAL := …;
      SELF\Solid.volume_of: REAL := …;
      …
  END_ENTITY;

  ENTITY Polyhedron
    SUBTYPE OF(Solid);
      faces: SET [4:?] OF Face;
      edges: SET [6:?] OF Edge;
      vertices: SET [4:?] OF Vertex;
    DERIVE
      euler_number: INTEGER := SIZEOF(vertices)+SIZEOF(faces)-
SIZEOF(edges);
```

```
      …
    WHERE
      Euler_rule: (euler_number <= 2) AND (euler_number MOD 2
= 0);
  END_ENTITY;
…
END_SCHEMA;


MAP_SCHEMA SetTheoreticGeometry;
  SOURCE ConstructiveSolidGeometry;
  TARGET ConstructiveSolidGeometry;

  MAP CSGOperation ABSTRACT SUPERTYPE OF (ONEOF(CSGUnion,
CSGIntersection, CSGSubtraction));
      tolerance: REAL;
    IN
      operand1: Solid;
      operand2: OPTIONAL Solid;
    OUT
      result: Solid;
    DERIVE
      may_intersect: BOOLEAN :=
IntersectBoxes(operand1.bounding_box, operand2.bounding_box);
      …
  END_MAP;

  MAP CSGSubtraction
    SUPERTYPE OF (ONEOF (Box_Subtraction,
Half_Space_Subtraction))
    SUBTYPE OF(CSGOperation);
    FORWARD CallCSGSubtraction(SELF);
    PRE_CONDITION CallCSGSubtractionPrecondition(SELF);
    POST_CONDITION CallCSGSubtractionPostcondition(SELF);
    PRE_ESTIMATE CallCSGSubtractionPreestimate(SELF);
    POST_ESTIMATE CallCSGSubtractionPostestimate(SELF);
  END_MAP;

  MAP Box_Subtraction
    SUBTYPE OF(CSGSubtraction);
    IN
      SELF\CSGOperation.operand1: Box;
      SELF\CSGOperation.operand2: OPTIONAL Box;
    FORWARD …
    PRE_CONDITION …
```



```
    POST_CONDITION …
    PRE_ESTIMATE …
    POST_ESTIMATE …
  END_MAP;

  MAP Half_Space_Subtraction
    SUBTYPE OF(CSGSubtraction);
    IN
      SELF\CSGOperation.operand2: OPTIONAL Half_Space;
    FORWARD …
    PRE_CONDITION …
    POST_CONDITION …
    PRE_ESTIMATE …
    POST_ESTIMATE …
  END_MAP;
…
END_MAP_SCHEMA;
```

2.3.1. Принцип А. Наследование отображений

Будучи классифицированными, сами отображения могут быть представлены типами объектов в рамках рассмотренной выше объектно-ориентированной метамодели. Частичный порядок π на *Map* отражает иерархию типов отображений. Определим наследование отображений как механизм, регулирующий права преемственности для атрибутов, связей, пред-, постусловий и оценок качества в базовых и наследуемых отображениях, аналогично операции наследования, обычно применяемой в рамках объектно-ориентированного подхода. Атрибуты и связи интерпретируются как данные, инкапсулируемые отображениями, условия и оценки качества — как их методы.

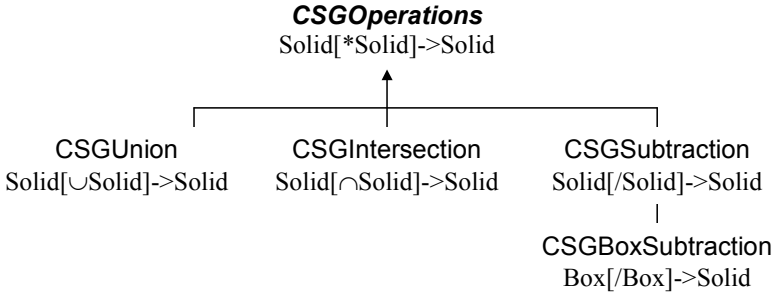

Рис. 3. Пример иерархии отображений для выполнения операций CSG

В примере на языке EXPRESS объявление схемы `ConstructiveSolidGeometry` вводит примитивы типа `Solid`, такие как `Sphere`, `Cube`, `Cylinder`, `Cone`, `Torus`, `Polyhedron`, `Box`, `Half_Space`, используя конструкцию `ENTITY`. Полиморфные методы `is_bounded`, `bounding_box`, `area_of`, `volume_of`,

boundary_representation объявляются в абстрактном супертипе Solid как вычисляемые (DERIVE) атрибуты и наследуются конкретными типами. Правила WHERE определяют условия, при которых экземпляры типов ENTITY являются семантически верными с точки зрения непротиворечивости значений их явных и вычисляемых атрибутов. Например, условие Euler_rule налагает известное ограничение Эйлера для непротиворечивого представления полиэдральных тел типа Polyhedron.

2.3.2. Принцип B. Структурная специализация и поведенческий полиморфизм

В схеме отображений SetTheoreticGeometry, использующей модель ConstructiveSolidGeometry посредством объявлений SOURCE и TARGET, определяются отображения типа CSGOperation, такие как CSGUnion, CSGIntersection, CSGSubtraction, реализующие теоретико-множественные операции над произвольными твердотельными геометрическими примитивами и их сборками. Фрагмент иерархии отображений, иллюстрирующий выполнение данных операций, представлен на рисунке 3. Кроме атрибута tolerance, соответствующего численной точности выполняемой операции, абстрактное отображение CSGOperation определяет входные связи operand1, operand2 и выходную связь result, соответственно, отмечаемые метками разделов IN и OUT. Семантически входные связи имеют отношение к операндам операции конструктивной твердотельной геометрии (CSG), а выходная связь — к ее результату. Опциональный статус, назначенный связи operand2, показывает, что второй операнд не является обязательным и может быть пропущен без потери целостности данных отображения. Допуская более содержательную реализацию на уровне конкретных операций, могут быть также определены специализированные (наследуемые) отображения. Например, отображение Box_Substraction специализирует отображение CSGSubstraction в результате наследования его свойств. Одновременно оно переопределяет входные связи путем объявления их типа как Box вместо Solid и предусматривает более эффективные реализации методов выполнения отображения, вычисления пред-, постусловий, оценок качества, выделяемых в теле отображения ключевыми словами FORWARD, PRE_CONDITION, POST_CONDITION, PRE_ESTIMATE, POST_ESTIMATE. Таким образом, в результате наследования отображений могут уточняться типы их атрибутов и связей, а также полиморфно переопределяться их поведенческие свойства.

2.3.3. Принцип C. Агрегатные отображения

Иногда в приложениях необходимо иметь возможность последовательно применить совпадающие или очень близкие по сигнатурам функций преобразования и объединить конечные результаты в соответствии с некоторой процедурой. Для этого родственные отображения могут быть сгруппированы с использованием агрегатных конструкций BAG, SET, ARRAY

или LIST, синтаксически подобных языковым конструкциям EXPRESS. Следующий пример иллюстрирует возможности определения агрегатных отображений. Заметим, что данный пример выглядит похожим на спецификацию EXPRESS, за исключением того, что CSGOperation и CSGOperationList являются типами отображений, а не типами данных.

```
TYPE CSGOperationList = LIST [1:?] OF CSGOperation;
END_TYPE;

LOCAL
  Union_Op: CSGUnion;
  Intersect_Op: CSGIntersection;
  Subtract_Op: CSGSubtraction;
  Complex_Op: CSGOperationList := [Union_Op, Intersect_Op,
Subtract_Op];
  solid1, solid2: Polyhedron;
  resulting_solids: LIST[1:?] OF Polyhedron;
END_LOCAL;

Complex_Op.operand1 := solid1;
Complex_Op.operand2 := solid2;
Complex_Op.result := resulting_solids;
Complex_Op.FORWARD();
```

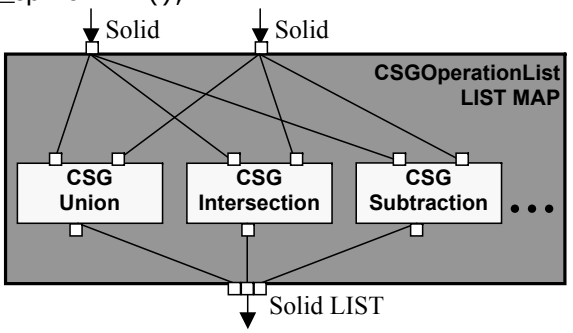

Рис. 4. Пример агрегатного отображения для множественного выполнения операций CSG

Семантика объявления агрегатного отображения соответствует определению производного отображения, основанного на назначенном типе. Каждый экземпляр агрегатного отображения представляет собой контейнер экземпляров основного отображения, связи которых непосредственно соединяются со связями агрегатного отображения. Это означает, что все экземпляры отображений, включенные в агрегатное, совместно используют его общие связи. Выполнение агрегатного отображения состоит в последо-вательном выполнении агрегируемых экземпляров основного отображения в порядке, предопределенном заданным агрегатным типом языка EXPRESS и его

допустимым интервалом размера. Агрегатное отображение может получать доступ к данным через входные связи, обновлять их через смешанные связи и накапливать выходные результаты путем последовательного выполнения отдельных экземпляров агрегируемых отображений. Типы входных и смешанных связей агрегатного отображения совпадают с соответствующими типами основного отображения. Типы выходных связей являются заданными EXPRESS агрегациями выходных связей его основного отображения. Таким образом, агрегатное отображение совместно использует общие связи с вложенными отображениями и объединяет их частные поведенческие свойства.

В вышеприведенном примере экземпляр `Complex_Op` агрегатного отображения `CSGOperationList` объединяет экземпляры `Union_Op`, `Intersect_Op`, `Subtract_Op` конкретных отображений `CSGUnion`, `CSGIntersection`, `CSGSub-traction`.. Структура данного агрегатного отображения показана на Рис. 4. В результате выполнения экземпляр агрегатного отображения `Complex_Op` принимает на входе твердотельные геометрические операнды `solid1`, `solid2`, применяет теоретико-множественные операции над операндами и генерирует выходные результаты как агрегацию полиэдров `resulting_solids`. Существенно, что созданное агрегатное отображение может впоследствии применяться к другим наборам данных и их потокам.

2.3.4. Принцип D. Выборочные отображения

Выборочное отображение — это конструкция, схожая с агрегатным отображением, за исключением того, что для получения выходных результатов должно применяться только одно из базовых отображений. Критерии выбора могут быть различными в зависимости от целей приложения. Поэтому, кроме конструкции `SELECT`, семантически означающей, что любой из экземпляров включенных базовых отображений может быть применен для получения результатов, вводится конструкция `PRE_SELECT`, выполняющая выбор по априорным оценкам и конструкция `POST_SELECT`, выполняющая все базовые отображения и выбирающая наилучший результат среди полученных с использованием апостериорных оценок. Допускается, что для критериев выбора могут быть использованы и другие возможности, и поддерживаемый набор селективных конструкций может быть значительно расширен. Для представленных конструкций предлагаются типовые реализации оценок качества, которые могут быть приняты по умолчанию в случаях, когда программист не предоставляет собственную реализацию.

Реализация по умолчанию для априорных оценок состоит в определении максимального различия уровней специализации для формальных и фактических входных параметров отображения. При использовании такого критерия выбирается базовое отображение, наиболее близкое по сигнатуре функции фактическим параметрам. Это достаточно общий и конструктивный метод, поскольку он позволяет выбрать то отображение, которое является

наиболее подходящим и, следовательно, эффективным для конкретной комбинации входных параметров.

Для апостериорных оценок могут быть использованы другие методы, которые определяют степень специализации выходных результатов отображения и их соответствие целевой схеме. Базовое отображение выбирается как наилучшее, если все сгенерированные выходные результаты принадлежат целевой схеме и занимают самые низкие позиции в иерархии типов, соответствующие наиболее специализированным основным, сложным и объектным типам. Здесь пропущены подробности того, как можно алгебраически выразить итоговые оценки качества и как их можно вычислить, используя словарь метаинформации.

В следующем примере предполагается, что оценочные функции для отображений вычитания `CSGSubtraction`, `Half_Space_Subtraction`, `Box_Subtraction` выбраны по умолчанию. Тогда определенный тип отображения `Selecti-veSubtraction` содержит всю функциональность, связанную с совместным управлением собранными в нем экземплярами базовых отображений, их выбо-ром в соответствии с контекстом приложения и последующим выполнением. Заметим, что реализации функций для базовых отображений могут варьиро-ваться по уровню численной точности, эффективности, надежности, принимая во внимание, что алгоритмические реализации операций CSG над произволь-ными полиэдрами, политопами и каноническими твердыми телами сущес-твенно различаются. Тем не менее, будучи примененным, отображение `Selec-tiveSubtraction` выбирает и выполняет базовое отображение, наиболее подхо-дящее для конкретных связанных с ним CSG операндов, как показано на Рис. 5.

```
TYPE SelectiveSubtraction = PRE_SELECT (CSGSubtraction,
Half_Space_Subtraction, Box_Subtraction);
END_TYPE;

LOCAL
  Op1: CSGSubtraction;
  Op2: Half_Space_Subtraction;
  Op3: BoxSubtraction;
  Complex_Op: SelectiveSubtraction;
  solid1, solid2, solid3: Solid;
END_LOCAL;

Insert(Complex_Op, Op1, 1);
Insert(Complex_Op, Op2, 2);
Insert(Complex_Op, Op3, 3);
```

```
Complex_Op.operand1 := solid1;
Complex_Op.operand2 := solid2;
Complex_Op.result := solid3;
Complex_Op.FORWARD();
```

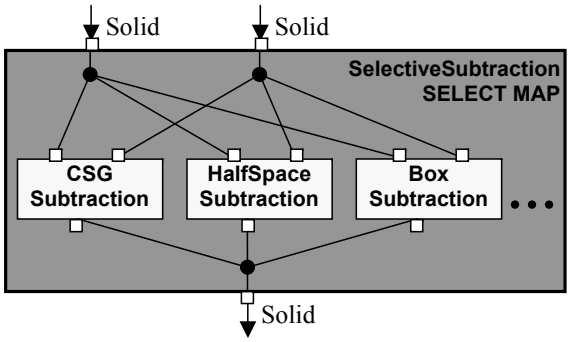

Рис. 5. Пример выборочного отображения для выполнения операций CSG в зависимости от специализации типов операндов

Для краткости, как и при обсуждении предыдущего принципа, не станем акцентировать внимание на возможных проблемах несоответствия связей базовых отображений, собранных внутри производных агрегатных и выборочных отображений. Единственное замечание состоит в следующем. Производные отображения могут использовать только те связи, которые явно встречаются в объявлении базовых отображений или их предков. Связи, определенные на более низких уровнях иерархии отображений, рассматриваются как необязательные и игнорируются. Связи, совместно используемые родственными базовыми отображениями, включаются в общий набор производных отображений. Таким образом, разрешается проблема несоответствия. Тем не менее, для наиболее общего случая композиции отображений подход предусматривает специальную конструкцию.

2.3.5. Принцип E. Зависимые отображения

Зависимое отображение — это тип обычных отображений, определяемых с использованием конструкции MAP. Ключевое отличие заключается в объявлении некоторых из его связей как экземпляров типа MAP. Причина поддерживать данный вид отображений состоит в необходимости параметризовывать реализации производных отображений другими базовыми отображениями.

Например, мы могли бы определить операцию CSGIntersection как отдельно реализованное отображение, наследуемое от абстрактного отображения CSGOperation с новой функциональностью для условий, оценок и функций отображения. Существует и другая возможность реализовать операцию CSGIntersection, используя уже определенное отображение CSGSubtraction.

В самом деле, CSG операция operand1∩operand2 может быть представлена как operand1\(operand1\operand2), и, следовательно, отображение CSGIntersection может быть реализовано как зависимое от CSGSubtraction. Для этого определение CSGIntersection включает объявление связи auxiliary_map с отображением CSGSubtraction и использует ее при вычислении отображения.

В примере, представленном на Рис. 6, для того, чтобы создать конкретный экземпляр отображения CSGIntersection, необходимо заблаговременно сконструировать экземпляр отображения CSGSubtraction и присвоить его этой связи. Существенно, что различные реализации зависимого отображения CSGIntersection могут быть получены в результате присвоения различных версий операции CSGSubtraction. Это можно легко сделать, не переписывая спецификации для данных отображений.

```
MAP CSGIntersection SUBTYPE OF(CSGOperation);
  INOUT
    auxiliary_map: CSGSubtraction;
  FORWARD …
END_MAP;
```

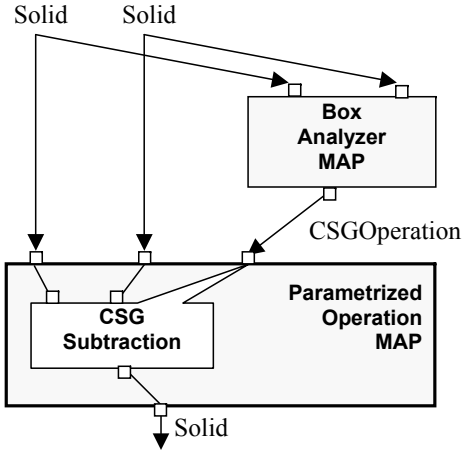

Рис. 6. Пример зависимого отображения для параметризации операций CSG

2.3.6. Принцип F. Составные отображения

Структурная компоновка более сложных отображений из уже определенных — один из основных принципов предлагаемого подхода. В соответствии с декларативными возможностями, основанными на расширениях языка EXPRESS, данный принцип может быть реализован посредством конструкции COMPOSITE_MAP. Подобно обычным отображениям, определяемым декларацией

MAP, конструкция COMPOSITE_MAP допускает явное объявление собственных атрибутов и связей. Но принципиальное различие заключается в способах того, как реализуются функции условий, оценок и отображений. Для обычных отображений реализации таких функций обеспечивает программист. Для составного отображения подобные функциональные реализации не используются, однако программист специфицирует его структуру как упорядоченное множество экземпляров взаимосвязанных друг с другом отображений. Будучи структурированными, составные отображения могут применяться как для прямых, так и для обратных преобразований данных.

В следующем примере составное отображение CSGAssemblyAnalysis предназначено для визуального представления пересекающихся твердых тел в виде сборки отдельных частей (см. рисунок 7). Данное отображение включает объявления внутреннего атрибута tolerance, определяющего точность проводимых вычислений, связей operand1, operand2, соответствующих входным операндам, и связей result1, result2, result3, через которые воспроизводятся результаты CSG операций operand1\operand2, operand1∩operand2, и operand2\operand1, соответственно.

```
COMPOSITE_MAP CSGAssemblyAnalysis;
  tolerance: REAL;
IN
  operand1: Solid;
  operand2: OPTIONAL Solid;
OUT
  result1: Solid;
  result2 : OPTIONAL Solid;
  result3 : OPTIONAL Solid;
STRUCTURE
  map1: SelectiveSubtraction;
  Op1: CSGSubtraction(tolerance);
  Op2: Half_Space_Subtraction(tolerance);
  Op3: BoxSubtraction(tolerance);
  Insert(map1, Op1, 1);
  Insert(map1, Op2, 2);
  Insert(map1, Op3, 3);
  map2: SelectiveSubtraction;
  …
  map3: SelectiveSubtraction;
  …
  map4: Transform3D;
  …
  map5: Transform3D;
  …
FLOW
  operand1 TO map1.operand1;
```



```
  operand2 TO map1.operand2;
  operand1 TO map2.operand2;
  operand2 TO map2.operand1;
  operand1 TO map3.operand1;

  map1.result TO map3.operand2;
  map1.result TO map4.operand;
  map2.result TO map5.operand;

  map3.result TO result2;
  map4.result TO result1;
  map5.result TO result3;
END_MAP;
```

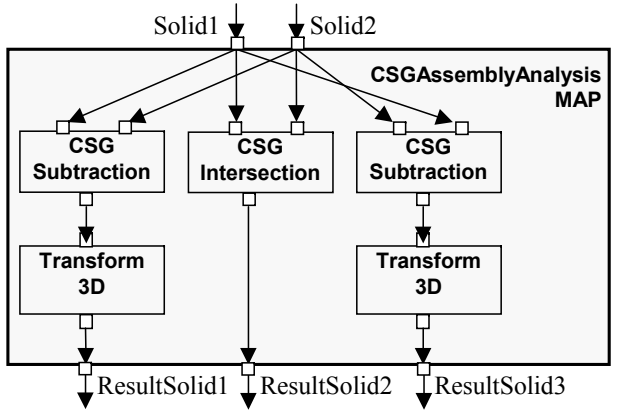

Рис. 7. Пример составного отображения для выполнения сложных операций CSG

Для этого отображение компонуется из отдельных базовых отображений, экземпляры которых конструируются в разделе, отмеченном ключевым словом STRUCTURE. Чтобы исключить неопределенность результатов, вызванную возможными циклическими зависимостями между базовыми отображениями, их экземпляры должны перечисляться в порядке, соответствующем фактической последовательности их выполнения. Экземпляры отображений связываются друг с другом в разделе FLOW посредством оператора TO, который присваивает выходные результаты одного преобразования входам другого и, таким образом, играет роль ветви потока данных, проходящих через общую структуру составного отображения.

Существуют простые правила, согласно которым базовые отображения могут быть однозначно взаимосвязаны и скомпонованы внутри производных отображений. Эти правила — следующие:

- связь, являющаяся источником данных (входные и смешанные связи составного отображения, а также смешанные и выходные связи базовых отображений), может быть присвоена связям, являющимся получателями данных (смешанные и выходные связи составного отображения, а также входные и смешанные связи базовых отображений);
- тип связи-получателя должен быть родственным (обобщенным или специализированным) типу связи-источника. Аналогично языку EXPRESS, специализация типов может охватывать соответствующие отношения между простыми типами данных, определяемыми пользователем объектными и созданными типами, а также накладывать дополнительные правила на соединение связей;
- одна и та же связь-источник может быть присвоена нескольким связям-получателям, но каждой связи-получателю может быть присвоена только одна связь-источник.

2.3.7. Принцип G. Композиционный сценарий

Наконец, представим сценарий, главное отображение, с которого должно начинаться выполнение всей схемы. Будем считать, что схема отображения может содержать только один сценарий. Сценарий включает встроенные входные и выходные связи `INPUT, OUTPUT` предопределенного типа `LIST[0:?] OF GENERIC`, объявления атрибутов, используемых в сценарии в качестве локальных переменных, раздел `STRUCTURE`, содержащий упорядоченный список экземпляров отображений и раздел `FLOW` для спецификации потока данных через составляющие отображения.

```
SCENARIO(INPUT,OUTPUT: LIST[0:?] OF GENERIC)

  STRUCTURE
    map1 : Map1(constructor);
    …
    mapN : MapN(constructor);
  FLOW
    INPUT TO map1.operand1;
    …
    mapN.result TO OUTPUT;
END_SCENARIO;
```

Входная связь `INPUT` используется для назначения сценарию набора входных данных, специфицируемых на языке моделирования EXPRESS и поставляемых в некотором мета-формате, а выходная связь `OUTPUT` используется для того, чтобы сохранить сгенерированные результаты в этом же представлении. Детали организации мета-формата относятся к вопросам реализации, обсуждаемым в следующем разделе.

Как и составные отображения, сценарий выполняется путем последовательного вызова функций отдельных отображений в порядке их перечисления и передачи входных, промежуточных и выходных данных от источников к получателям в соответствии с назначенными потоками данных. Тот же самый сценарий может использоваться как для прямого, так и для обратного отображения данных. Единственное требование состоит в том, чтобы все определенные в нем простые отображения были двунаправленными, то есть, для них должны быть заданы функции `FORWARD` и `BACKWARD`. В обратных преобразованиях роли входных и выходных связей, пред- и постусловий, пред- и постоценок, соответственно, меняются местами.

Таким образом, после того, как определен композиционный сценарий, нет необходимости указывать какие-либо еще процедуры, и программа отображения может быть немедленно выполнена.

3. Реализация подхода

Предложенный и проиллюстрированный выше подход допускает конструктивную реализацию в контексте семейства стандартов STEP (ISO 10303 — стандарт для представления данных об изделии и обмена этими данными) [5]. Язык моделирования данных EXPRESS является одной из частей (а именно, частью 11), определяемой и регулируемой в рамках STEP, который был разработан с целью описания моделей продуктов для таких отраслей промышленности как машиностроение, автомобилестроение, аэрокосмическая индустрия, судостроение, архитектура, нефте-газодобыча, электротехника и электроника. Согласно определениям STEP, EXPRESS — это язык абстрактных схем, который обеспечивает спецификацию объектов, принадлежащих некоторой предметной области, элементов информации, имеющих отношение к этим объектам, и накладываемых на них ограничений. Не являясь исполняемым языком, EXPRESS не предусматривает спецификацию организации и представления элементов информации. Эти вопросы регулируются другими частями STEP, определяющими мета-форматы обменных файлов (часть 21 — формат кодирования открытым текстом, часть 28 — XML формат), абстрактный API для работы со специфицированными на EXPRESS данными, называемый стандартным интерфейсом доступа к данным (SDAI, часть 22), а также ряд представлений SDAI на популярных языках программирования, таких как C++, C, Java (части 23, 24, 27).

Следовательно, разумно ожидать, что данный подход в сочетании с языком EXPRESS хорошо бы подошел для специфицирования визуализации сложных многопрофильных моделей продуктов в вышеупомянутых ключевых отраслях промышленности.

Нам видится несколько возможностей реализации композиционного подхода. Прежде всего, он может быть реализован с использованием представленных расширений языка EXPRESS, образующих новый язык спецификаций, называемый в дальнейшем EXPRESS-F. Являясь расширением EXPRESS, язык

EXPRESS-F позволяет специфицировать как сами данные, так и их потоки в контексте выполняемых отображений. Сочетая парадигмы объектно-ориентированного программирования и потоков данных, язык предусматривает различные возможности для спецификации простых отображений, а также для составления из них более сложных сценариев. Ранее парадигма потоков данных успешно применялась при разработке широкого класса приложений визуализации [6, 7], что дало основания использовать ее и в настоящей работе.

Спецификации отображений, написанные на языке EXPRESS-F, могут интерпретироваться непосредственно, а также могут транслироваться на популярные исполняемые языки. В этом смысле язык EXPRESS-F, как и язык EXPRESS, нейтрален по отношению к возможным платформам реализации.

Для построения программ на основе обсуждаемого подхода предложена и разработана инструментальная среда общего назначения. Данная среда — это система классов для представления произвольных модельно-ориентированных данных и отображений, для их интроспекции во время запуска с использованием словарей метаданных, а также для выполнения определяемых отображений и композиционных сценариев. Таким образом, общая функциональность среды включает:

- Управление метаданными, относящимися к исходной и целевой моделям данных и к схеме их отображения друг в друга;
- Экспорт/импорт модельно-ориентированных данных;
- Асинхронный доступ к совместно используемым данным через программные интерфейсы;
- Составление сценариев во время работы, проверку их непротиворечивости и выполнение;
- Решение задач геометрического моделирования с использованием специализированной библиотеки отображений.

Инструментальная среда совместима со SDAI, регламентированным стандартом STEP, и предоставляет все декларируемые в нем возможности для экспорта/импорта данных, специфицируемых на языке EXPRESS. Так, среда поддерживает все предусматриваемые SDAI классы, такие как сессия, транзакция, репозиторий, экземпляр схемы, модель, набор объектов, экземпляр приложения, а также все классы, необходимые для воспроизведения схемо-зависимых типов данных, основанных на иерархии исходных типов EXPRESS, показанной на Рис. 2. Последние определяются для версий раннего и смешанного связывания в рамках SDAI. В дополнение к типам данных, среда управляет представлениями отображений, специфицированных на языке EXPRESS-F. С этой целью она поддерживает иерархию классов отображений, изображенную на Рис. 8 и предусматриваемую языком EXPRESS-F.

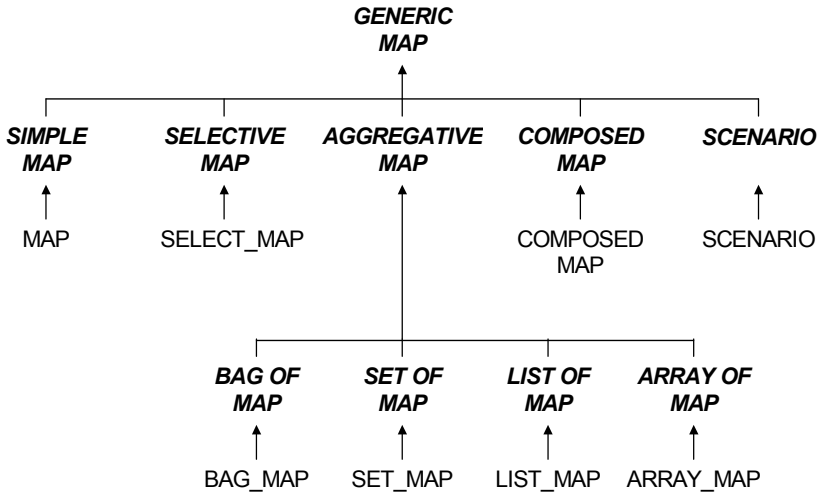

Рис. 8. Иерархия классов отображений инструментальной среды

Среда также использует и расширяет словари метаданных SDAI для интроспекции данных и отображений, а также предоставляет средства управления метаданными, включая загрузку схем, специфицированных на языках EXPRESS и EXPRESS-F, навигацию по словарям и конструирование объектов с использованием фабрик. Словари метаданных являются ключевыми компонентами ядра инструментальной среды, которые позволяют реализовать широкий набор вышеперечисленных функций.

Существенно, что составление сценариев, их проверка на непротиворечивость и выполнение могут совершаться в режиме реального времени, предоставляя конечным пользователям возможности как использовать встроенные сценарии, так и определять новые непосредственно во время сеанса работы с приложением.

Чтобы упростить разработку программных компонентов, типичных для большинства приложений визуализации, инструментальная среда предлагает готовые к использованию библиотеки отображений для решения задач геометрического моделирования. В частности, данные библиотеки реализуют преобразования над каноническими геометрическими примитивами и теоретико-множественные операции над произвольными полиэдрами.

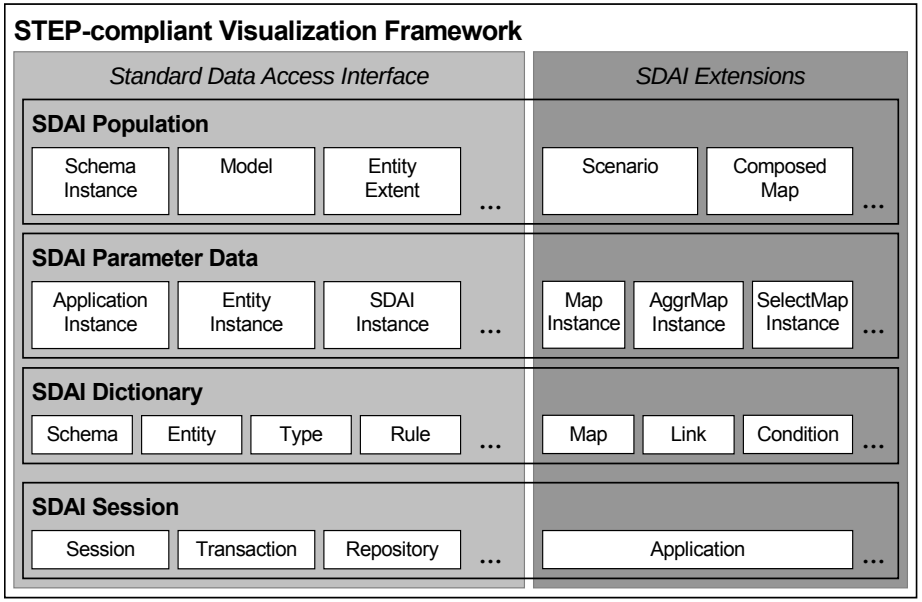

Рис. 9. Компонентная организация инструментальной среды

4. Приложение

Предложенный подход и разработанная среда использовались и тестировались при создании программного приложения, предназначенного для виртуального строительства. Приложение ориентируется на актуальные потребности строительной индустрии в средствах макетирования архитектурно-строительных проектов и моделирования процессов строительства с использованием технологий виртуальной реальности. Данные средства должны способствовать пониманию особенностей проекта между различными заинтересованными сторонами, включая заказчиков, архитекторов, инженеров, экономистов, менеджеров, принимающих участие в совместном проекте, с целью исключить возможные ошибки проекта еще на подготовительных этапах, устранить препятствия на пути его реализации и, в конечном итоге, улучшить сроки и качество его реализации.

Следуя данному подходу, при создании приложения виртуального строительства необходимо определить исходную и целевую информационные модели, надлежащим образом описывающие прикладную и визуальную предметные области. Использование в качестве моделей существующих промышленных стандартов позволило бы при этом обеспечить интероперабельность приложения и придать ему практическую ценность. В силу этих причин построенное приложение использует в качестве исходной модели международный информационный стандарт Industry Foundation Classes (IFC),

разработанный альянсом International Alliance for Interoperability (IAI) [8], а в качестве целевой модели — язык моделирования виртуальной реальности (VRML), спроектированный и реализованный консорциумом Web3D [9, 10].

4.1. Информационный стандарт IFC

Главная миссия IAI состоит в достижении интероперабельности между строительными компаниями во всем мире посредством предоставления универсальной основы для согласованного совместного использования информации. С момента основания IAI в 1995 году членство в нем возросло до более 600 компаний из более чем 20 стран. IAI продвигает спецификации IFC, которые позволяют компаниям по разработке программного обеспечения создавать новое поколение интероперабельных приложений для архитектурно-строительной индустрии. Сейчас IFC получил широкое распространение внутри этого сообщества.

IAI использует методологию STEP и язык EXPRESS для определения и реализации всеобъемлющей многопрофильной информационной модели, включающей в себя множество аспектов и дисциплин, образующих архитектурно-строительную индустрию.

В ходе разработки и согласования спецификаций под управлением IAI были подготовлены их отдельные версии: IFC 1.5.1, IFC 2.0, IFC 2x, IFC 2x2. Начиная с первой версии, спецификации IFC претерпевали существенные изме-нения, заключающиеся в совершенствовании структуры всей модели и в выделении таких предметных областей, как архитектура, электроаппаратура, отопление, вентилирование и кондиционирование воздуха, управление строительством, управление оборудованием. Высокая степень интеграции и сложность модели IFC при этом значительно возрастали. В таблице 1 приведена некоторая статистика, показывающая динамику увеличения числа используемых типов данных и наложенных на них ограничений для модели IFC. По-видимому, подобная динамика продолжится и в ожидаемой версии IFC 3.0.

Версия IFC	IFC 1.5.1	IFC 2.0	IFC 2x	IFC 2x2
Созданные типы	59	103	139	202
Определенные типы	36	54	89	110
Типы объектов	186	290	370	623
Ограничения	122	168	207	293
Общее количество типов и ограничений	403	615	805	1228

Таблица 1. Статистика использования исходных типов и ограничений в семействе стандартов IFC

4.2. Информационный стандарт VRML

В течение последнего времени язык моделирования виртуальной реальности (VRML) получил распространение в ряде отраслей науки, образования и промышленности, а также признан Internet сообществом в качестве стандартного языка для описания интерактивных трехмерных миров внутри всемирной паутины. Текущие версии стандарта (VRML97 и X3D) предусматривают расширенный репертуар объектов для трехмерного геометрического моделирования и мультимедиа, развитые поведенческие механизмы, улучшенные форматы кодирования и программные интерфейсы приложения. Являясь расширением VRML97, версия X3D добавляет известные возможности, такие как новые типы сенсоров, геопространственные объекты, гуманоиды, NURBS-формы, поддержку формата XML, интерфейс авторизации сцен и профиль для совместимости с MPEG4.

Поскольку как VRML97, так и X3D модели определяются неформальным методом, для данных стандартов были разработаны EXPRESS схемы с тем, чтобы согласовать исходную и целевую модели, а также унифицировано управлять ими в рамках инструментальной среды. Фрагмент иерархии объектных типов в рамках схемы, разработанной на языке EXPRESS для информационной модели VRML97, представлен на Рис. 10.

- VRML97Node
  - ChildNode
    - BindableNode
      - BackgroundNode
        - Background
      - Fog
      - NavigationInfo
      - Viewpoint
    - GroupingNode
      - Anchor
      - Billboard
      - Collision
      - Group
      - LOD
      - Switch
      - Transform
    - LightNode
      - DirectionalLight
      - PointLight
      - SpotLight

  - GeometryNode
    - Box
    - Cone
    - Cylinder
    - ElevationGrid
    - Extrusion
    - IndexedLineSet
    - IndexedFaceSet
    - PointSet
    - Sphere
    - Text
  - SensorNode
    - EnvironmentalSensorNode
      - ProximitySensor
      - VisibilitySensor
    - PointingDeviceSensorNode
      - TouchSensorNode
        - TouchSensor
      - DragSensorNode
        - CylinderSensor
        - PlaneSensor
        - SphereSensor
    - KeyDeviceSensorNode

Рис. 10. Фрагмент иерархии объектных типов модели VRML97

Подробные спецификации разработанных на языке EXPRESS схем для информационных моделей VRML97 и X3D могут быть найдены на странице http://www.ispras.ru/~step.

4.3. Основанный на отображениях сценарий для виртуального строительства

Очевидно, что визуализация такой многогранной информации, как IFC данные, в VRML-совместимой среде виртуальной реальности является нетривиальной задачей и требует для реализации применения специального программного подхода. Принимая во внимание сложность моделей IFC и VRML, а также необходимость применения разнообразных методов и сценариев визуализации, использование предлагаемого подхода представляется достаточно перспективным. Дополнительная мотивация использования данного подхода заключается в том, что довольно часто сценарий должен формироваться пользователем непосредственно во время работы, чтобы визуально воспроизвести те аспекты, которые важны для решаемой задачи на текущей

стадии анализа данных. Разработанная инструментальная среда предоставляет для этого необходимые средства, что является ее серьезным преимуществом.

Для применения предлагаемого подхода и инструментальной среды следует разработать сценарии, значимые для решения проблем виртуального строительства. Рассмотрим некоторые их примеры, типичные для приложений визуализации.

4.3.1. Задача A (Назначение визуальных свойств)

Чтобы отобразить архитектурно-строительный проект наиболее эффектным и исчерпывающим способом, пользователи часто используют физически содержательные метафоры и назначают различным типам архитектурных, отделочных, электрических, транспортных и других элементов материалы, визуальные свойства которых соответствуют их реальным физическим свойствам. Фрагмент иерархии типов элементов, встречающихся в моделях IFC, представлен на Рис. 11. Естественно, поскольку коллекция таких элементов может насчитывать сотни типов, пользователи не всегда способны различить назначенные материалы и адекватно идентифицировать элементы. Возможное решение состоит в назначении материалов широким категориям элементов в соответствии с предпочтениями пользователей. Но в этом случае необходимо разработать сценарий, согласно которому следует выбирать тот материал, который в наилучшей степени соответствует конкретному типу элемента.

- IfcRoot
  - IfcObject
    - IfcProduct
      - IfcElement (Material1)
        - IfcBuildingElement (Material2)
          - IfcWall (Material3)
            - IfcWallStandardCase (Material4)
          - IfcColumn
          - IfcBeam
          - IfcSlab
          - IfcWindow (Material5)
          - IfcDoor
          - IfcStair
          - IfcStairFlight
          - IfcRamp
          - IfcRampFlight
          - IfcRoof
          - IfcBuildingElementProxy
        - IfcOpeningElement
        - IfcFurnishingElement
        - IfcElectricalElement (Material6)
        - IfcTransportElement
        - IfcEquipmentElement
        - IfcDistributionElement

Рис. 11. Фрагмент иерархии объектных типов модели IFC и назначенные материалы

Эта процедура может быть реализована путем использования выборочного отображения `IfcMaterialMap`, которое основывается на предопределенном отображении `AbstractMaterialMap` и его производных `StandardMaterialMap`, `CADMaterialMap`.

```
MAP AbstractMaterialMap ABSTRACT SUPERTYPE OF (ONEOF
(StandardMaterialMap, CADMaterialMap … ));
  IN
    element: IfcElement;
  OUT
    material: Material;
END_MAP;

MAP StandardMaterialMap SUBTYPE OF AbstractMaterialMap;
  ascribed_element: STRING;
  PRE_ESTIMATE: return LEVEL_DIFF(ascribed_element,
TYPEOF(element));
END_MAP;

TYPE UserMaterialMap = PRE_SELECT (StandardMaterialMap);
END_TYPE;

TYPE IfcMaterialMap = PRE_SELECT (CADMaterialMap,
UserMaterialMap);
END_TYPE;
```

Априорная оценка для отображения `StandardMaterialMap` связана с вычислением разницы в уровнях специализации типов фактических элементов, поступающих в экземпляр отображения через входную связь element, и типов элементов `ascribed_element`, которым назначены материалы. Напомним, что отображение `UserMaterialMap` функционирует как селектор экземпляров `StandardMaterialMap` по критерию, реализуемому функциями оценок. Это означает, что в соответствии с приведенным на Рис. 11 назначением материалов, для каждого конкретного элемента отображение выберет визуальное свойство, наиболее подходящее по его типу. Например, для стандартных стен будет выбран `Material4`, для нестандартных — `Material3`, для окон — `Material5`, для других строительных элементов

(колонн, балок, перекрытий) — `Material2`, для электрических элементов — `Material6` и `Material1` — для всех остальных категорий элементов.

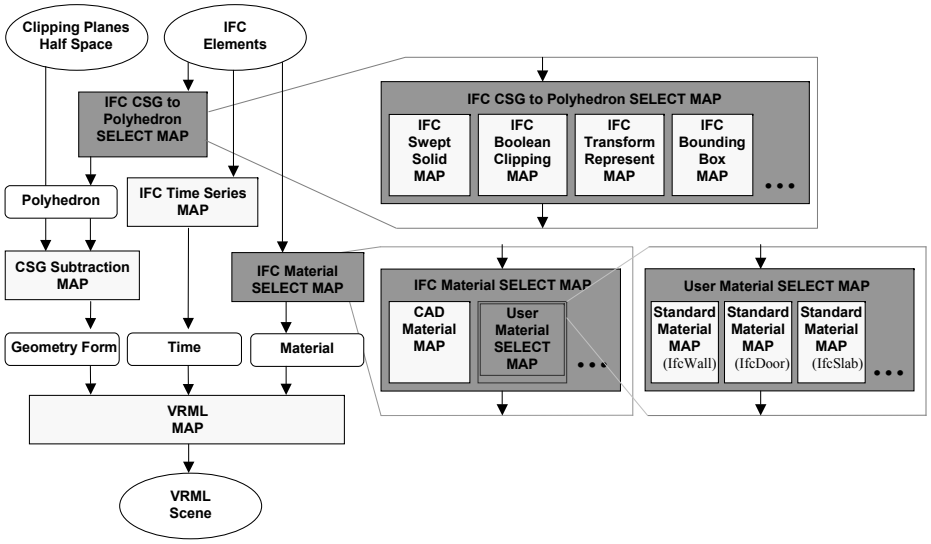

Рис. 12. Сценарий виртуального строительства с использованием IFC данных

Назначение визуальных свойств может показаться тривиальной проблемой, которая могла бы решаться путем применения полиморфных методов в иерархии IFC классов. Но это не совсем так по следующим причинам. Прежде всего, невозможно предвидеть все вероятные приложения и случаи использования, в которых могут участвовать IFC данные. Поэтому иерархия IFC типов должна реализовываться в виде библиотеки, ответственной за представление данных, но не за их обработку. Вторая причина заключается том, что пользователю необходимо назначать визуальные свойства динамически во время сеанса работы приложения, и, следовательно, они не могут быть жестко встроены в реализацию классов. Третья причина состоит в многообразии критериев выбора, которые могут не совпадать с анализом типов в рамках имеющейся иерархии наследования и должны быть реализованы особым способом.

Например, в сценарии, представленном на Рис. 12, селектор материала `UserMaterialMap` участвует в другой, внешней, процедуре выбора, основанной на отображении `MaterialMap`. Оно комбинирует техники выбора с использо-ванием отображения `CADMaterialMap`, которое назначает материалы тем элементам, которым уже были приписаны некоторые визуальные свойства, например, в ходе работы CAD приложения. Таким

образом, вложенные выборочные отображения допускают гибкую реализацию более сложных и практически значимых сценариев визуализации.

4.3.2. Задача B (Обработка альтернативных геометрических моделей)

Следующая задача, которую можно считать типовой для приложений визуализации, связана с использованием альтернативных геометрических моделей для представления элементов промышленного проекта, например, кон-структивной твердотельной модели или модели граничного представления. В ряде случаев, элементы могут задаваться несколькими моделями одновре-менно. Это означает, что следует реализовать некоторую стратегию преобра-зования и выбора моделей. Как и в предыдущем случае, такая стратегия должна определяться пользователем и может изменяться при выполнении приложения.

Например, IFC 2x элементы могут геометрически представляться как параллелепипеды, тела, образованные движением контура или отсечением, тела, заданные граничным представлением, а также как объекты, представленные одной из перечисленных выше моделью с дополнительно наложенными геометрическими преобразованиями. Несколько моделей может использоваться одновременно с учетом разнообразия приложений и задач, решаемых пользователями. В представленном сценарии отображение `IfcCSGtoPolyhedronMap` реализует стратегию преобразования и выбора моделей с использованием базовых отображений `IfcSweptSolidMap`, `IfcBooleanClippingMap`, `IfcTransformedRepresentationMap`, `IfcBoundingBoxMap`, соответствующих различным геометрическим моделям. При подобной реализации пользователь может управлять сценарием обработки и визуализации моделей.

С другой точки зрения, часто степень детализации предоставленных геометрических данных и сложность их преобразований могут быть чрезмерно высоки, чтобы обеспечить возможность интерактивной работы пользователя. В таких случаях критерии качества базовых преобразований могут быть переопределены в соответствии со сложностью необходимых вычислений.

Базовые преобразования определяются как простые специфицируемые программистом отображения. Так, согласно Рис. 13, отображение `IfcSweptSolidMap` извлекает из IFC данных геометрическую модель тела, образованного движением контура, и генерирует в качестве выхода полиэдр. Для этого оно реконструирует контур и траекторию движения, представленные каноническими геометрическими формами, с использованием отображений `IfcProfileMap` и `IfcExtrusionMap` соответственно. Последние выбирают и применяют экземпляры отображений `ExtrudedCurveMap`, `RevolvedCurveMap`, и `CompositeProfileMap`, `RectangleProfileMap`, `CircleProfileMap` в зависимости от фактических типов геометрических данных. Отображение `SweptSolidToPolyhedronMap`,

используя кривые контура и траектории, генерирует выходное геометрическое представление в виде полиэдра.

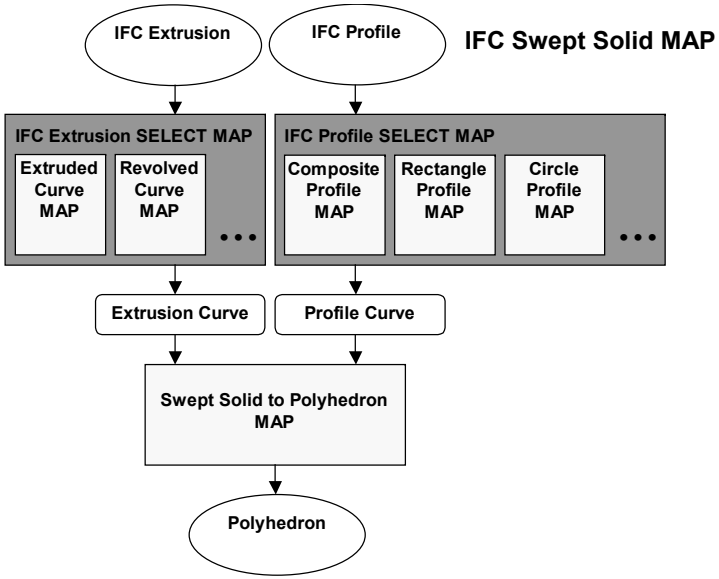

Рис. 13. Пример простого отображения для визуализации твердых тел, полученных движением контура по заданной траектории

Другой пример реализации простого отображения `IfcBooleanClippingMap` приведен на Рис. 14. Оно предназначено для вычитания полупространства из твердого тела, представленного одной из описанных выше геометрических моделей. Его реализация основана на использовании предопределенных отображений `IfcCSGtoPolyhedronMap` и `IfcHalfSpaceMap`. Важно отметить, что зависимые отображения `IfcCSGtoPolyhedronMap` и `IfcBoolean-ClippingMap` используют друг друга для преобразования рекурсивно определяемой геометрической модели IFC в каноническую полиэдральную форму. Аналогичным образом реализуется отображение `IfcTransformed-RepresentationMap`, которое выполняет стандартные трехмерные преобразования над геометрическими данными, представленными в одной из описанных выше альтернативных форм. Таким образом, отображения могут использовать друг друга для обработки сложных рекурсивно определяемых геометрических моделей.

Вернемся к сценарию виртуального строительства, представленному на Рис. 13. Данный сценарий содержит также отображение `CSGSubtractionMap` для отсечения визуализируемого здания с целью предоставить лучшую панораму деталей проекта, обычно скрываемых вложенными архитектурными формами и конструкциями. В сценарии отсекающие плоскости и

полупространства опционально определяются пользователем. Рис. 15 и 16 представляют примеры использования таких методик.

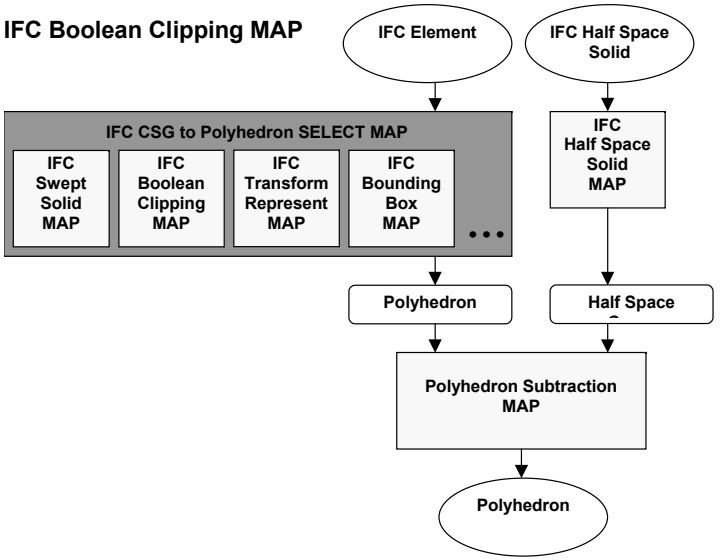

Рис. 14. Пример реализации простого отображения для визуализации отсеченных твердых тел

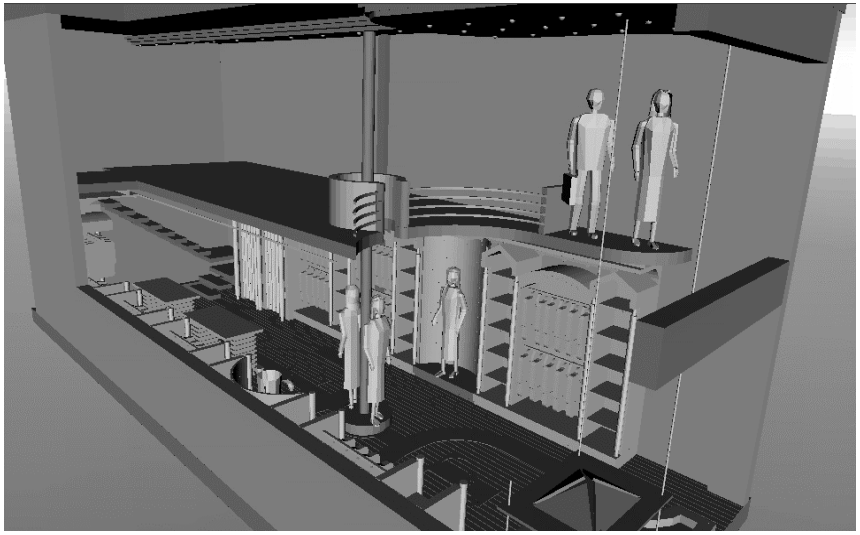

Рис. 15. Пример визуализации проекта здания с использованием отсекающих плоскостей

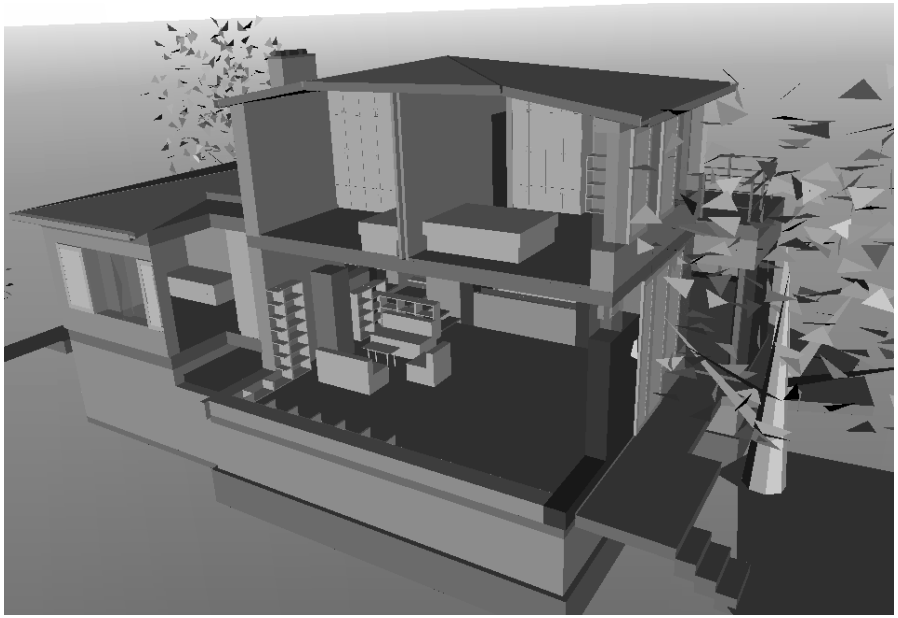

Рис. 16. Пример визуализации проекта здания с использованием полупространственных отсечений

Сценарий может анимировать процесс виртуального строительства путем использования соответствующих IFC данных о проекте. С этой целью отображение `IfcTimeSeriesMap` анализирует последовательность действий в рамках строительного проекта и назначает соответствующим элементам временные интервалы, в течение которых они должны быть включены в визуализируемую сцену. Отображение `VRMLMap` группирует геометрические формы, материалы и временные интервалы, полученные отображениями сценария, и генерирует результаты в виде статических или динамических VRML сцен. Рис. 17 показывает последовательность кадров, демонстрирующую участникам проекта, как должен осуществляться процесс строительства, какие недостатки и препятствия могут потенциально возникнуть при его реализации и что следует сделать, чтобы устранить их еще на предварительных стадиях.

Таким образом, рассмотренные сценарии могут эффективно использоваться при решении проблем виртуального строительства. Данные сценарии применялись при разработке IFC2VRML конвертера, предназначенного для моделирования процессов проектирования и строительства зданий с использованием технологии виртуальной реальности. Приложение может быть загружено через Internet со страницы http://www.ispras.ru/~step.

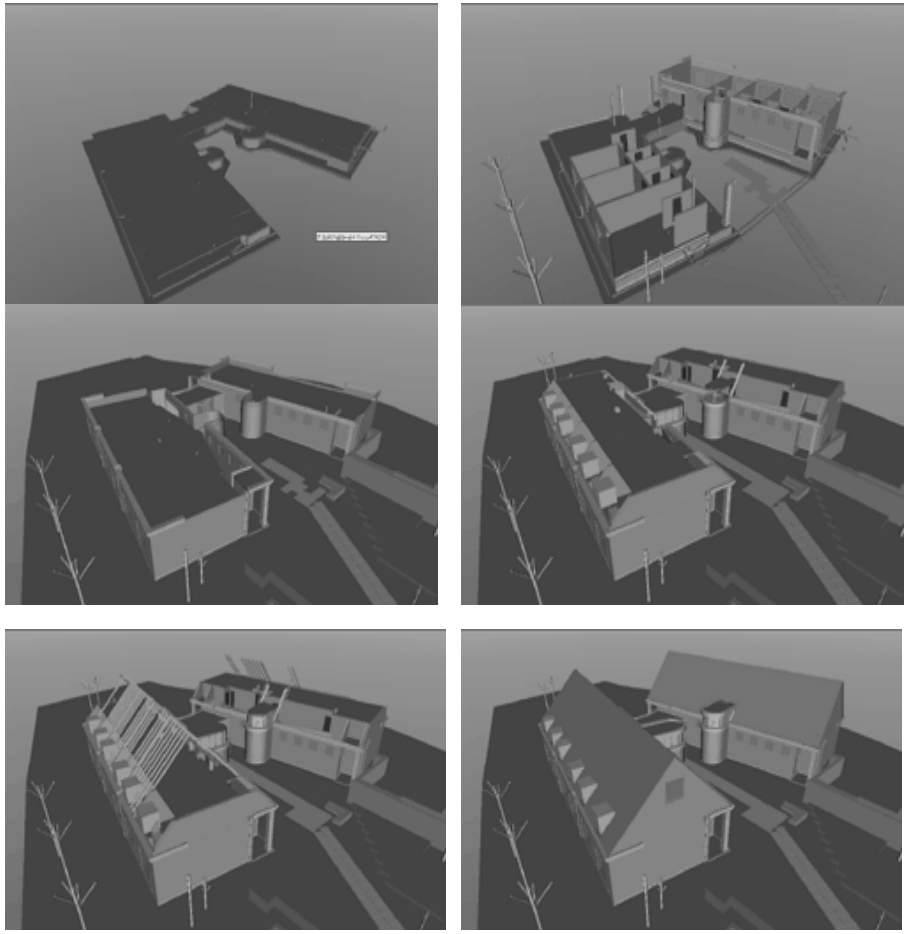

Рис. 17. Пример последовательности кадров фильма, демонстрирующего процесс виртуального строительства

5. Заключение

Проведенные исследования показывают, что композиционный программный подход может успешно применяться при построении приложений визуализации, к которым предъявляются повышенные требования гибкости, надежности, многократного использования. В частности, его использование представляется перспективным для приложений, оперирующих со сложной междисциплинарной информацией на основе сценариев, определяемых и настраиваемых пользователями непосредственно в ходе рабочей сессии.

Подход допускает эффективную реализацию с использованием предложенного декларативного языка потоков данных EXPRESS-F и описанной инструментальной среды разработки приложений визуализации общего назначения. Ориентация на актуальные международные стандарты представления промышленной информации STEP и моделирования виртуальных миров VRML/X3D делает представленные программные решения естественным выбором при построении приложений визуализации промышленной продукции с использованием технологий виртуальной реальности в среде Интернет/Интранет.

Разработанное приложение для виртуального макетирования архитектурно-строительных проектов доказывает состоятельность и осуществимость предлагаемого подхода, а также обуславливает его внедрение в другие предметные области науки и промышленности.

Литература

1. D.A. Duce, D.J.Duke. Interaction, Cognition and Visualization. In P.Palanque, R.Bastide, editors, Proceedings of Eurographics Workshop on Design, Specification andVerification of Interactive Systems, Springer-Verlag,Wien, NewYork,1995, pp.1-20.
2. W.Felger, M.Fruhauf, M.Gobel, R Gnatz, G.R.Hofmann. Towards a Reference Model for Scientific Visualization Systems. In M.Grave, Y.Le Lous, W.T. Hewitt, editors, Proceedings of Eurographics Workshop on Visualization in Scientific Computing, Springer-Verlag,Wien, NewYork,1994, pp. 63-74.
3. R.Breu, U.Hinkel, C.Hofman, C.Klein, B.Paech, B. Rumpe, and V. Thurner. Towards a Formalization of the Unified Modeling Language // In ECOOP'97 — 11th European Conference on Object-Oriented Programming, Proceedings, ed. M.Aksit and S.Matsuoka, LNCS 1241, Springer-Verlag, 1997, pp.344-366.
4. M.Richters and M.Gogolla. On Formalizing the UML Object Constraint Language OCL // In ER'98 — 17th Int. Conf. Conceptual Modeling, Proceedings, ed. Tok-Wang Ling, LNCS 1507, Springer, 1998, pp. 449-464.
5. ISO 10303: Industrial Automation systems and integration — Product data representation and exchange. ISO 10303-11:1994, Description methods: The EXPRESS language reference manual. ISO 10303-21:1994, Implementation methods: Clear text encoding of the exchange structure. ISO 10303-22:1998, Implementation methods: Standard data access interface. ISO 10303-28:2003, Implementation methods: XML representations of EXPRESS schemas and data.
6. Klimenko S., Nikitin I., Burkin V., Semenov V., Tarlapan O., Hagen H. Visualization in string theory. // ACM Computers & Graphics, vol. 24, num. 1, 2000, pp.23-30.
7. Semenov V., Morozov S., Tarlapan O., Belyaeva A. Component-based development of scientific computing applications in OpenMV environment. // Proceedings of 16th IMACS World Congress, Lausanne, Switzerland, August 21–25, 2000, ed. Deville M., Owens R., ISBN 3-9522075-1-9.
8. Industry Foundation Classes, Release 2. The International Alliance for Interoperability, 1999, URL: *http://www.iaiweb.lbl.gov*.
9. The Virtual Reality Modeling Language. International Standard ISO/IEC 14772-1:1997.
10. Web3D Consortium, URL: *http://www.web3d.org*.