УДК 004.422

УСЛОВНАЯ ВИЗУАЛИЗАЦИЯ ПРОСТРАНСТВЕННЫХ СЦЕН С ИСПОЛЬЗОВАНИЕМ ДИНАМИЧЕСКОГО ГРАФА

Горбацевич Ф. Ф., Кудрявцев А. С., Шмидт В. К.

Санкт-Петербургский Государственный Электротехнический Университет
E-mail: gorbatsevich@gmail.com

Аннотация. Рассматривается расширение графа сцены для визуализации моделей сцен с условным отображением, в рамках компьютерной графики. Предлагается подход условного обхода графа при визуализации модели сцены. Производится сравнение предложенного подхода с существующими организациями графа сцены.
Ключевые слова: компьютерная графика, динамические сцены, граф сцены.

Введение

Системы визуализации с использованием компьютерной графики в настоящее время используются в широком спектре областей. Моделирование геопространственной (в том числе подводной) обстановки – одна из таких областей. Система отображения и наблюдения (СОН) была создана нами в рамках работы над тренажерным комплексом экипажа подводной лодки. При создании подобной системы возникает проблема отображения модели подводной сцены; сцена включает подводные лодки, корабли и иные динамические и статические объекты. Одним из требований являлось отображение информации в различных видах, в зависимости от режима просмотра (реалистичный, упрощенный, картографический). Для каждого из этих режимов выдвигались различные требования к представлению объектов (триангуляция, пиктограмма), наличию окружения (реалистичное, упрощенное), выделению подпространства (сота вокруг активного объекта), наличию вспомогательных элементов (линии отстояния от дна).

На сегодняшний день большинство моделей сцен описываются при помощи статических графов сцены, с фиксированным поведением узлов. Под поведением мы понимаем выполняемые узлом действия, когда обход графа затрагивает узел. В виду большого количества моделируемых объектов и сложности поведения в современных моделях, это приводит к тому, что в графе описывается только часть взаимотношений объектов, а остальное поведение реализуется в рамках описания поведения, содержащейся в узлах графа, и выполняемого, когда обход затрагивает узел (примером такого поведения является отображение вспомогательных объектов и т.п., в зависимости от условий наблюдения в модели: точки наблюдения и других) [1].

Предлагается создать динамический граф сцены с описанием поведения модели на декларативном языке программирования. В этом случае сокращается объем императивного кода, так как часть поведения модели, реализованная как описание поведения в узлах графа, становится меньше.

Получаемое таким образом декларативное описание определяет возможные условия визуализации (например, свойства точки наблюдения), реакции на условия, влияющие на наличие объектов в сцене, их внешний вид и поведение.

Существующие подходы к организации графа сцены

Существующие подходы описывают трехмерные сцены в виде дерева, которое формально представляется орграфом без контуров:

$T = \langle V, E \rangle$,

где V – множество вершин, а E – подмножество дуг $V \times V$, при этом все вершины V равнозначны и все представляют объекты сцены. Введем обозначение, что дуги представляют родительско-дочерние отношения между объектами; и в графе нет контуров. Также, множество исходящих из узла дуг упорядочено, для того чтобы обход происходил в строгом порядке. Строгий порядок необходим для однозначности действия коммутирующих узлов при обходе.

Подобная структура модели называется иерархической объектно-ориентированной [5]; в настоящий момент она применяется в моделях сцен чаще, чем альтернативные структуры [2, 3] (плоские, пространственно-ориентированные и другие) [5].

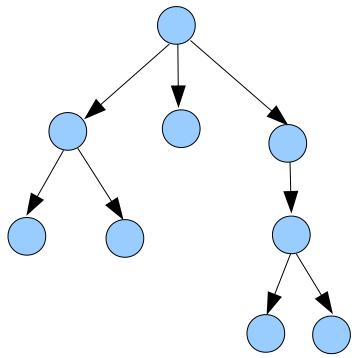

Рис. 1. Традиционный граф сцены: узлы-объекты и отношения между объектами. Упорядоченные дуги позволяют обходить граф слева направо.

Недостатком такого подхода для решения нашей задачи является необходимость осуществлять проверку необходимых условий визуализации при обходе узлов. Учитывая требования к частой перемене вида объектов от режима наблюдения, можно предложить иной подход, описывающий большую часть поведения в организации самого графа.

Визуализация в рамках подобной модели сцены осуществляется обходом графа, при котором каждый посещённый узел выводится на экран [4].

Такой обход можно назвать потоком управления, или путём следования сигнала.

Динамический граф сцены

Предлагаемая нами модель сцены имеет следующее отличие: в графе множество вершин неоднородно и состоит из вершин различных типов:

$V_o \cup V_c \cup V_a \cup V_d \cup V_i = V$,

V_o – вершины-объекты, V_c – вершины-коммутаторы (коммутирующие узлы), V_a – вершины-активаторы, V_d – деактиваторы, V_i – вершины-условия. Узлы-коммутаторы влияют на обход дочерних подграфов узла, в зависимости от условий, активированных или деактивированных в соответствующих узлах.

Граф имеет вид

$T = <V, E, R>$,

здесь E – множество дуг, $E \subseteq V \times V$, граф $T_1 = <V, E>$ не содержит контуров; R – множество дуг $R \subseteq V \times V$, граф $T_2 = <V, R>$ может содержать контуры. Таким образом, граф T более не является деревом, поскольку в нём возможны контуры. Узлы V_a,V_d и V_i рассматриваются ниже, в разделе «обход графа».

Подобная модификация имеет своей целью условный обход графа, что в конечном счёте позволяет при визуализации объектов переключать представление объектов в зависимости от условий наблюдения. Условность обхода графа обусловлена тем, что конечный результат отображения сцены (визуализированное изображение, представление

визуализированных объектов) зависит от начальных условий обхода графа для заданного кадра визуализации.

Граф назван динамическим из-за условности обхода графа, что позволяет визуализировать динамические сцены при различных условиях наблюдения; структура самого графа постоянна.

Родительско-дочерние отношения E определяют порядок обхода (и, следовательно, порядок визуализации), в то время как ссылки R являются вспомогательными связями между узлами графа.

Максимальная степень коммутирующего узла v не превышает двух, □(v) □ 2, то есть число исходящих дуг из коммутирующего узла не превышает двух.

Определим, что коммутирующий узел может иметь внутреннее поведения. В зависимости от поведения обход, совершаемый из этого узла по дугам, проходит только по одному из дочерних подграфов узла (подграфом, соединённым с данным узлом одним из дуг, исходящих из узла). Это приводит к тому, что обход графа является условным.

Пример графа с коммутирующим узлом $v \subset V_c$ показан на рис. 2. В зависимости от некоторого условия (или внутреннего поведения условного узла), визуализация проходит либо по левому, либо по правому поддереву коммутирующего узла. Это позволяет создавать объекты с различным представлением, в зависимости от условий наблюдения.

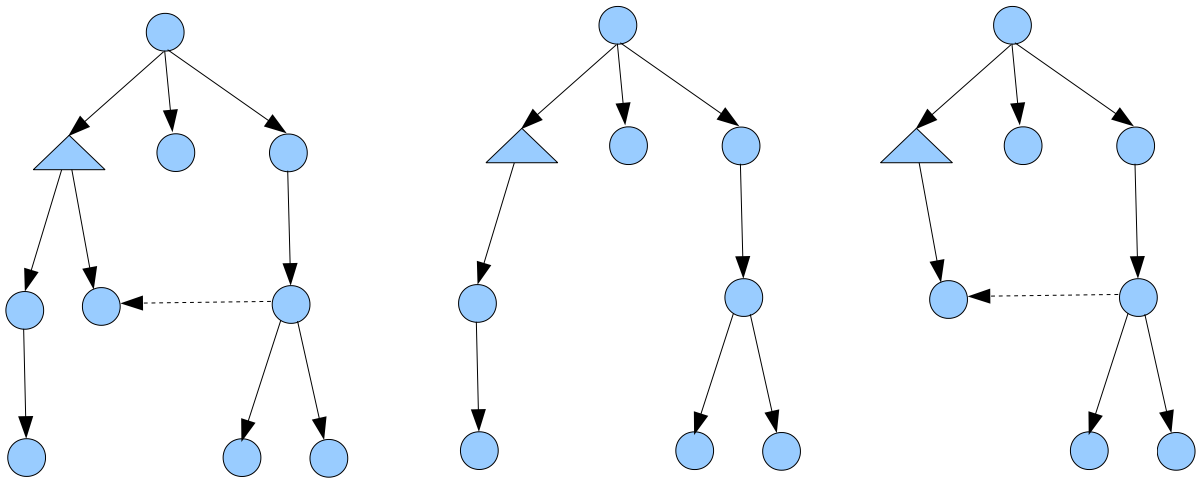

Рис. 2. Граф с коммутирующим узлом, дуги-ссылки показаны пунктиром:
(а) исходный вид, (б) подграф посещенных узлов при одном срабатывании коммутирующего узла, (в) при другом

Обход графа

Как правило, для получения конечного изображения, необходимо несколько раз обойти дерево с различными целями, для вычисления различных аспектов сцены, таких, как освещение, отсечения, визуализация и т.п. Поэтому необходима возможность иметь несколько сигналов (видов обхода графа), которые будут запускаться на графе последовательно. Различные виды сигнала можно рассматривать как разноцветные фишки, передвигаемые по узлам графа.

Таким образом, предлагаемый граф можно рассматривать как преобразование множества данных D_0 в множество данных D_n при помощи модификации данных при обходе динамического графа. Исходными данными D_0 являются начальные данные о сцене, конечными – изображение на экране.

Поскольку граф изменяется, в зависимости от внутренних состояний коммутирующих

узлов, к каждому из проходов подграф посещаемых узлов G_k будет отличаться от подграфа посещенных узлов G_{k-1}; здесь $G_k \subset G$, $1 \le k \le n$, где n – число видов сигнала (число обходов графа в каждом кадре).

Если определены правила преобразования подграфа посещаемых улов G_0 в G_{n-1}: $G_0 \rightarrow G_1 \rightarrow G_2 \rightarrow ... \rightarrow G_{n-1}$, то преобразования множеств данных можно записать следующим образом:

$$D_0 \xrightarrow{G_0} D_1 \xrightarrow{G_1} D_2 \rightarrow ... \xrightarrow{G_{n-1}} D_n .$$

Подграф G_{k-1}, $1 \le k \le n$, в своих узлах ссылается на элементы и подмножества множества данных D_k, а также содержит правила преобразования элементов множества и правила трансформации в подграф G_k. Последовательное применение правил преобразования графов позволит преобразовывать множество данных.

Поясняя на примере, D_0 — это начальные сведения о сцене для каждого кадра при отображении в определённом окне, а D_n — это конечное визуализированное изображение кадра в этом окне. Множества данных с D_1 по D_{n-1} представляют из себя внутренние состояния.

Пример последовательности сигналов: сигнал обработки координатно-временной информации, сигнал рассчёта отсечения, сигнал выбора степеней детализации, сигнал расчёта освещения, сигнал визуализации. Так, для выбора степени детализации из числа дискретных представлений объекта можно ввести коммутирующий узел, перенаправляющий сигнал в один из своих дочерних подграфов в зависимости от расстояния до наблюдателя.

Множество сигналов и их последовательность и будут определять набор преобразований $G_0 — G_n$.

Передача сигнала в узлы, достижимые из коммутирующего узла выполняется (или не выполняется), в зависимости от условий, которые адресуются в коммутирующем узле.

Условием «прозрачности» узла для прохождения сигнала может быть как проверка вида сигнала (например, расчёт теней не выполняется для объектов, не имеющих представления, поэтому сигнал визуализации дальше не пропускается), или проверка активности узла-условия. Рассмотрение узла-условия подразумевает также наличие узлов-активаторов и узлов-деактиваторов.

Узел-условие $v \subset Vi \subset V$ — специальный вид узла, который приводится в активное состояние (получает значение «истина»), когда сигнал проходит узел-активатор, ссылающийся на данный узел-условие. Соответственно, он приводится в неактивное состояние, когда сигнал проходит узел-деактиватор, ссылающийся на данный узел-условие.

Пример вида графа с узлом-условием показан на рис. 3: объект А будет отображён только в том случае, если во время обхода графа посещён узел-активатор: условие активируется, и коммутирующий узел становится прозрачным для сигнала.

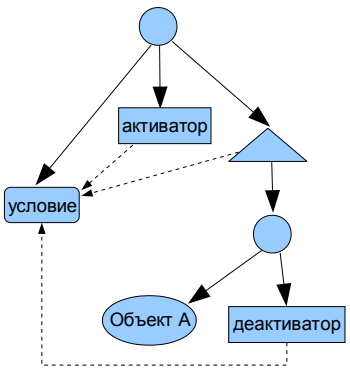

Рис. 3. Узел-условие, активатор, коммутирующий узел и деактиватор.

Кроме потока управления, рассмотренного выше, можно также выделить *поток*

данных. Под потоком данных мы понимаем получение, передачу и использование данных, которое происходит в графе, с учётом дуг-ссылок.

Параллелизм в графе возможен на двух уровнях: распараллеливание сигнала при его проходе по графу, и параллельное выполнение нескольких сигналов. В первом случае необходимо выявление зависимостей узлов в графе, и последующее увеличение числа обрабатывающих граф программных потоков в узлах, имеющих более одного дочернего узла, учитывая выявленные зависимости. Для параллельного выполнения проходов нескольких сигналов необходимо выявление зависимости данных в графе.

Предлагается следующий алгоритм обхода графа:

```
<function parse(signal, node)>
    <if type(node) is node_commutator>
        <then>
    <if type(node) is node_activator>
        <condition = get_condition(node)>
        <condition = True>
        <for child of ordered_children(node)>
    <parse (signal, child)>

    <if type(node) is node_deactivator>
        <condition = get_condition(node)>
        <condition = False>
        <for child of ordered_children(node)>
    <parse (signal, child)>

    <if type(node) is node_commutator>
        <condition = get_condition(node)>
        <if condition == True>
        <then parse(get_child(node, 1), signal)
        <if condition == False>
        <then parse(get_child(node, 2), signal)

    <else>
    <if is_transparent(node, signal)>
        <for child of ordered_children(node)>
    <parse (signal, child)>
```

Приведенный алгоритм обходит граф за время с верхней оценкой $O(n \cdot s)$, где n – число элементов в графе, а s – число сигналов (с учётом того, что подмножество исходящих вершин известно для каждой вершины).

Применение в системе отображения и наблюдения

Декларативное описание графа сцены выполняется на XML (eXtensible Markup Language), и загружается программой отображения в иерархическую структуру данных, после чего совершаются обходы графы, что приводит к визуализации сцены для каждого кадра.

Объекты в системе хранятся в DOM (document object model), начальный вид графа сцены загружается из XML.

Граф реализуется набором ООП-объектов, содержащих ссылки на родительский и дочерние объекты, представляющие дуги из множества E потока управления.

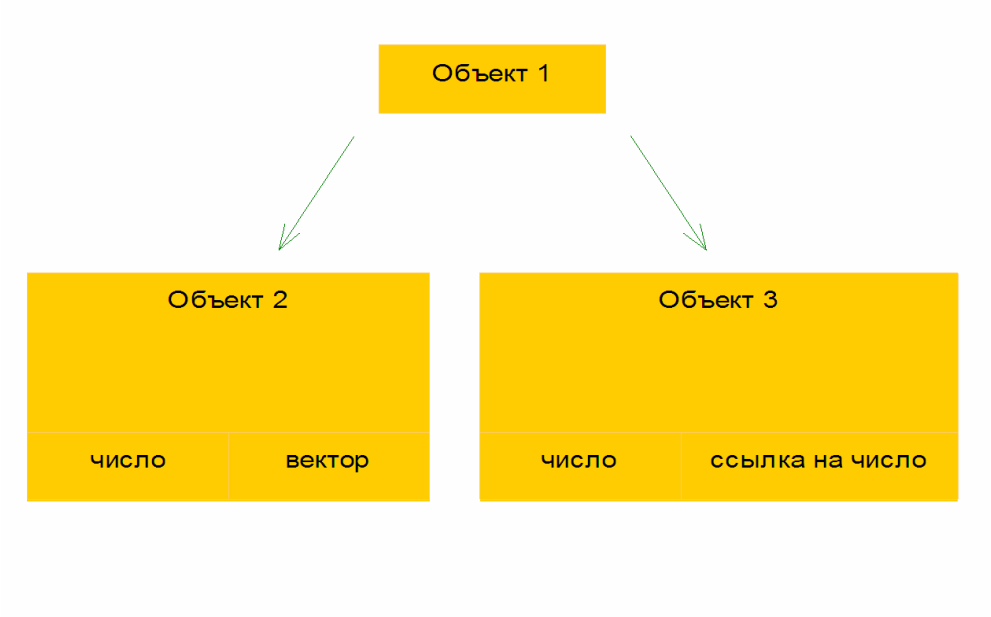

Рис. 4. Реализация ссылок как зависимых атрибутов

Реализованы средства для рефлексии и интроспекции ООП-объектов для сериализации DOM в XML, а также отображения свойств объектов в пользовательском интерфейсе. Для создания ссылочного механизма реализован класс «ссылка», позволяющий ООП-объекту ссылаться как на другой ООП-объект (элемент в графе), так и на атрибут другого ООП-объекта, рис. 4. В последнем случае реализован механизм зависимых атрибутов (атрибут не имеет собственного значения, и всегда возвращает заимствованное значение). Подобный механизм существует в ряде других программных каркасов, в том числе в декларативном описании пользовательских интерфейсов XAML (eXtensible Application Markup Language) в технологии WPF (Windows Presentation Foundation) от Microsoft [6].

Система реализована на C++, на базе собственного графического ядра, с использованием графического интерфейса OpenGL.

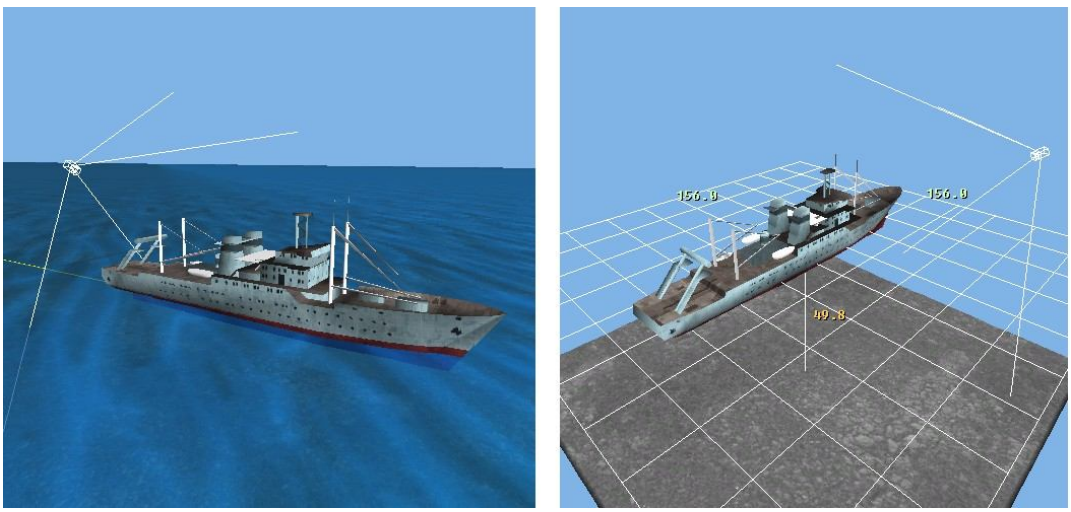

Рис. 5. Реалистичное отображение и отображение с вспомогательными построениями

Ниже даны примеры описания декларативного описания на XML. В графе существуют описания:

окон (view)

условий, активизируемых этими окнами (condition),

точек наблюдения в трехмерном пространстве (camera).

```
<rootobject>
    <views>
        <view rCamera="CameraFlight" />
        ...
    </views>
    <conditions>
        <condition sName="CameraIsFlight"/>
        ...
    </conditions>
    <scene sName="Scene">
        <camera sName="CameraFlight" mMatrix="...">
            <activator rCondition="CameraIsFlight" />
            ...
```

Применение разработанного графа позволило получить в созданной системе [7]:

Отображение в нескольких различных видах (реалистичный, картографический, вспомогательный).

```
<sobject sName="Helicopter1">
        <fork rCondition="CameraIsFlight">
            <tri_object sTriFile="heli.model" mMatrix="..." />
        </fork>
        <fork rCondition="CameraIsMap">
            <icon_object sIconFile="heli.bmp" mMatrix="..." />
        </fork>
</sobject>
```

Отображение объекта в зависимости от условий наблюдения (триангулированное представление или пиктограмма, в зависимости от расстояния до наблюдателя).

Отображение окружающей среды.

Отображение вспомогательных объектов (траектории, линии отстояния от дна и от уровня моря, пирамида видимости камеры, размерные числа и т.п.).

Выделение соты интересующего объёма из общего объёма сцены (рис. 5).

Сравнение с существующими графами сцены

К преимуществам предлагаемого подхода можно отнести:

Большая гибкость алгоритма, возможность произвольно расширять поведение модели, изменяя граф, для визуализации с расчетом различных параметров (освещения, отсечения), в отличие от необходимости переопределения внутреннего поведения в подходе с фиксированным графом.

Упрощение разработки за счет декларативного описания динамики взаимодействия объектов сцены.

К недостаткам относятся:

Меньшая эффективность по сравнению с традиционными подходами, обход графа занимает больше времени.

Менее оптимизированное для скорости визуализации решение, по сравнению с подходами, направленными на буферизацию графических данных (при помощи вертексных буферов и т.п.).

Выводы

Динамический граф сцены позволяет эффективно описывать условную визуализацию динамической пространственной сцены. Применение оправдано в трехмерных приложениях, требующих различных видов представлений объектов. Как правило, это приложения-симуляторы, ГИС и т.п.

Предложенный подход был применен при реализации системы отображения и наблюдения (СОН) подводной сцены для ФГУП «Комета», используемой для тренировки экипажей подводных аппаратов, рис. 5. Программная система СОН позволяет следить за ходом тренировки экипажа подводной лодки, производя измерения параметров прохождения подводной лодки (линии отстояния, траектории, расстояния до объектов).

Литература

1. H. Sowizral. Scene Graphs in the New Millennium // IEEE Computer Graphics and Application, Volume 20 , Issue 1. 2000.
2. J. Dollner and K. Hinrichs. A Generalized Scene Graph // Proceedings of Vision, Modeling and Visualization. IOS Press, Amsterdam, 2000
3. Li et al. High-extensible scene graph framework based on component techniques // J Zhejiang Univ SCIENCE A 2006 7(7):1247-1252
4. Foley D., van Dam A. и др. Computer Graphics: Principles and Practice. Addison-Wesley, 1997. 1174 с.
5. Chang A. A Survey of Geometric Data Structures for Ray Tracing // Technical Report TR-CIS-2001-06. CIS Department, Polytechnic University, 2001.
6. Troelsen A. Pro C# 2008 and the .NET 3.5 Platform // Apress, 2007.
7. Горбацевич Ф.Ф, Кудрявцев А.С., Шмидт В.К. Система отображения и наблюдения пространственных динамических сцен для тренажерных комплексов. // Материалы 58-й научно-технической конференции профессорско-преподавательского состава ЛЭТИ. СПбГЭТУ, 2005.