Программное средство визуализации графических сцен на основе WebGL*

Коршунов С.А., Николайчук О.А., Павлов А.И.
grey.for@gmail.com, nicoly@icc.ru, asd@icc.ru
Иркутск, Россия, Институт динамики систем и теории управления имени В.М. Матросова (ИДСТУ СО РАН)

Данная работа описывает основные принципы функционирования предлагаемого авторами программного средства визуализации графических сцен. Оно позволит непрограммирующим специалистам-предметникам создавать собственные подсистемы визуализации в различных областях применения, а использование онтологий даст им возможность в работе оперировать исключительно понятиями исследуемой области.
Изложена архитектура программного средства и отдельных модулей, а также модели онтологий, используемые в работе. Рассмотрен вопрос использования программной библиотеки стандарта WebGL для визуализации и приведены примеры ее использования для создания графической модели промышленной зоны.

Keywords: *визуализация, онтология, трехмерное моделирование, машинная графика.*

1. Введение

Одним из самых перспективных направлений компьютерной графики является трехмерное моделирование, т.е. создание 3D-объектов различной сложности и направленности [2,3,8,11]. На данный момент трехмерная визуализация применяется практически в любой сфере человеческой деятельности. Объекты научных исследований становятся все более сложными и разнообразными и требуют адекватных программных средств визуализации для наглядного отображения своих результатов [9]. В образовательном процессе визуализация применяется для создания трехмерных образовательных проектов, таких как простые анимированные обучающие материалы или сложные виртуальные лаборатории, которые позволят повысить интерес учащихся и скорость усвоения ими учебного материала [4,10] и т.д.
При этом во многих областях применения преобладают узконаправленные средства визуализации, не предоставляющие пользователю полноценного инструментария для создания моделей, выходящих за рамки предметных областей, на которые ориентированы эти приложения. Все они характеризуются ограниченным набором встроенных объектов и эффектов, а разработка собственной подсистемы визуализации может потребовать от пользователя значительных навыков в программировании. Все это обуславливает необходимость разработки такого программного средства, которое предоставило бы пользователям инструментарий для создания собственных визуальных подсистем различного применения, а автоматизация процесса написания кода снизила бы порог вхождения для непрограммирующих пользователей.
Цель данной работы – описание основных принципов разработки подобного программного средства, на основе программной библиотеки стандарта WebGL [6].

2. Архитектура программного средства визуализации

Формально опишем программное средство визуализации в виде теоретико-множественных операций:

$$VT = \left\langle R^O, R^S, T, G, V, Ont^D, Ont^{VO}, Ont^{AppV} \right\rangle, \quad (1)$$

где VT – программное средство визуализации,
R^O – редактор графических объектов,
R^S – редактор графических сцен,
T – транслятор правил описания сцены,
G – модуль генерации кода объектов и сцены,
V – модуль визуализации,
O_{nt}^D – онтология предметной области,
O_{nt}^{VO} – онтология визуальных объектов,
O_{nt}^{AppV} – онтология визуализации приложения.
Результатом работы программного средства будет являться сформированная графическая сцена, которая в совокупности с браузером для ее отображения, входными данными и библиотекой стандарта WebGL образует подсистему визуализации:

$$VSS = \left\langle GS, DI, BR, GL \right\rangle, \quad (2)$$

где VSS – подсистема визуализации,
GS – графическая сцена,
DI – входные данные, определяющие динамику ее поведения,
BR – браузер для отображения сцены,
GL – программная библиотека для создания трехмерной графики в браузере.

Работа частично поддержана грантами РФФИ №16-37-00122, № 16-07-05641 и № 15-37-20655.Работа опубликована по гранту РФФИ №16-07-20482

В соответствии с вышеприведенным описанием мы можем сформировать архитектуру программного средства (рис. 1).

Опишем основные функциональные компоненты:

- Редактор графических объектов, позволяющий пользователю создавать свои графические объекты разной степени сложности, используя в качестве основы геометрические примитивы (прямоугольник, сфера, цилиндр и т.д.), либо полигональные сетки. Для хранения описания объектов предлагается использовать онтологию графических объектов.
- Редактор графических сцен, предназначенный для описания структуры и поведения графической сцены в виде набора логических правил [5]. Описание сцен также предполагается хранить в соответствующей онтологии.
- Транслятор, преобразовывающий набор логических правил в структуру данных.
- Модуль визуализации, позволяющий на основе структуры данных, полученной от транслятора, сформировать трехмерную графическую модель, соответствующую моделируемой области и содержащую визуальные объекты, созданные пользователем. В качестве средства для отображения трехмерной визуальной модели предлагается использовать браузер.
- Модуль генерации кода графической сцены и ее объектов. Генерация кода происходит на основе структуры данных, получаемой модулем от транслятора. Сгенерированный код является отчуждаемым и может использоваться вне программного средства при помощи любого браузера.

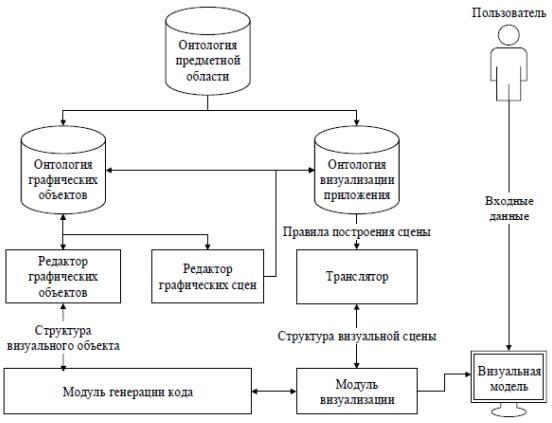

Рис. 1: Архитектура программной системы.

Для реализации редакторов графических объектов и моделей авторы предлагают использовать программную библиотеку стандарта WebGL, позволяющую создавать интерактивную графику на языке JavaScript, отображаемую в веб-браузере. Данное решение означает, что у пользователя не будет необходимости в установке какого-либо стороннего программного обеспечения. Библиотеки данного стандарта могут быть задействованы в широком спектре совместимых с ней браузеров и предоставляет достаточно возможностей для создания графических объектов и моделирования динамики их поведения.

Для хранения графических объектов, а также правил описания их поведения и компоновки в графические сцены предлагается использовать иерархию онтологий:

- Онтология графических объектов, содержащая созданные пользователем визуальные объекты, которые впоследствии могут быть использованы в качестве основы для создания новых объектов, путем дополнения или изменения их структуры.
- Онтология визуализации приложения, содержащая структуры визуальных сцен, а именно, то, какие объекты будут находиться на сцене и правила отображения этих объектов.
- Онтология предметной области, содержащая классы и понятия исследуемой области.

При отображении моделей онтологии визуальных объектов и онтологии приложения на модель хранения данных получаем ее структуру (рис. 2).

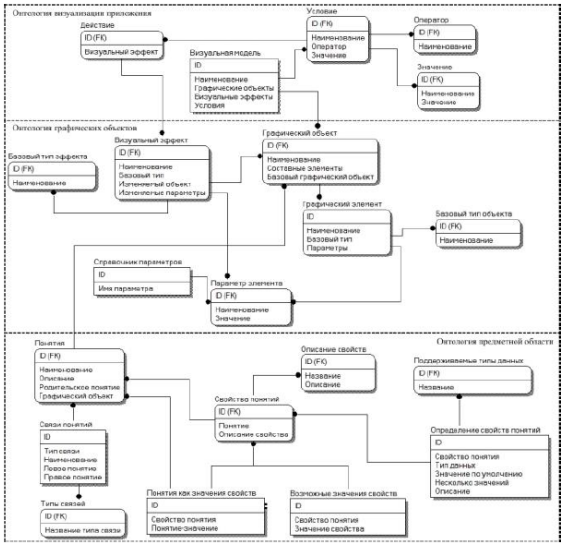

Рис. 2: Логическая модель онтологий.

Согласно данной модели, каждому объекту предметной области будет соответствовать визуальный объект, наиболее полно отображающий все его основные свойства. В свою очередь, каждый визуальный объект может поддерживать несколько различных визуальных эффектов (анимаций), которые могут изменять определенные свойства объекта. Каждый экземпляр анимации содержит ссылку на ее тип и на атрибуты/свойства объекта, которые будут изменены в процессе анимации.

3. Особенности реализации

Рассмотрим этапы технологии визуализации на примере модели промышленной площадки, состоящей из ряда объектов (промышленное здание, угольный террикон, дорога и несколько транспортных средств).

3.1 Создание графических объектов

Первым этапом технологии визуализации является создание трехмерных объектов с достаточным уровнем детализации соответствующих реальным объектам. Для этих целей предназначен редактор графических объектов, созданный на основе программной библиотеки стандарта WebGL, достаточно простой в использовании для непрограммирующих пользователей и позволяющий при этом создавать объекты самой разной структуры.

Для сочетания простоты использования редактора и сложности создаваемых объектов авторы предлагают использовать геометрические примитивы и полигональные сетки в качестве основы для объектов (рис. 3). Геометрические примитивы легко использовать, так как большинство реальных объектов можно разложить на множество более мелких элементов стандартных форм (куб, цилиндр, сфера, плоскости различной формы и т.д.), также объекты на основе примитивов можно использовать в тех случаях, когда для модели не требуются объекты с повышенной детализацией (рис. 3а). Использование полигональных сеток позволит более точно настроить форму объекта, а также дополнять простые объекты через добавление незначительных деталей (рис. 3б). Сочетание двух этих подходов позволит создавать трехмерную визуализацию с достаточной степенью детализации для большинства визуализируемых моделей.

Опциональным дополнением редактора является импорт объектов наиболее распространенных форматов (3ds, VRML, X3D) [5].

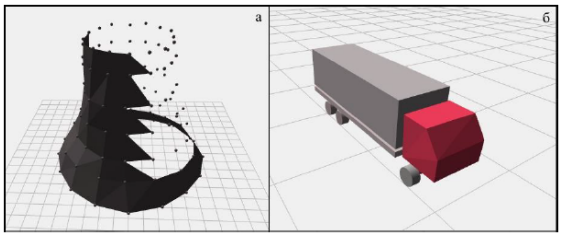

Рис. 3: Примеры графических объектов: на основе полигональных сеток (а); на основе сочетания геометрических примитивов (б).

4.1 Формирование правил построения сцен

После создания набора графических объектов требуется скомпоновать их в единую графическую сцену. Для этого с помощью редактора графических сцен необходимо описать их расположение на сцене, а также поведение, в виде правил. Правила отражают зависимость описания графической сцены от различных параметров (входных данных, полученных от пользователя, параметров других объектов).

Динамика всей сцены (т.е. ее объектов) определяется правилами поведения, которые описывают операции над объектами – изменение их свойств, структуры, добавление/удаление со сцены. Условием для срабатывания любого случая может быть значение свойства объекта, элемента или входного параметра.

Приведем описание правил построения сцены для рассматриваемого примера (рис. 4):

```
Simple «Build_10_r» placement (-470 0 0) (0 90 0)
Extend «Road_20» placement (-420 0 250) (-420 0 -600)
parameter «length»
Simple «Coil_3» placement (-350 -10 -50) (0 270 0)
Simple «Truck_test_1» placement (420 -5 150) (0 180 0)
Simple «Truck_test_2» placement (420 -5 50) (0 180 0)
Simple «Truck_test_3» placement (420 -5 150) (0 180 0)
If «t» >= 0 then change_object_parameter
«Truck_test_1» «z» 1 500
If «t» >= 10 then change_object_parameter
«Truck_test_2» «z» 1 550
If «t» >= 20 then change_object_parameter
«Truck_test_3» «z» 1 600
```

Данное описание указывает на размещение на графической сцене нескольких объектов различной структуры: «Build_10_r» и «Coil_3» (промышленное здание и угольный террикон) и дорожное полотно «Road_20».

В рассматриваемом примере три правила описывают перемещение объектов «Truck_test» по оси Z в зависимости от значения входного параметра «t», который может быть параметром какого-либо уже созданного объекта, либо может передаваться от источника данных. В совокупности, данная структура данных описывает графическую сцену.

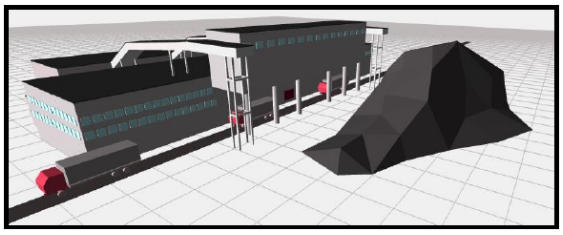

Рис. 4: Пример графической сцены.

Если условно разделить работу программного средства на два этапа, подготовительный, направленный на описание модели, т.е. создание объектов и формирование правил их поведения, и основной, обеспечивающий создание подсистемы визуализации, то первый этап является пользовательским, а второй – системным, реализуемым автоматически.

Созданное описание графической сцены схраняется в онтологии визуализации приложения.

5.1 Генерация программного кода

Созданные пользователем структура данных и правила, соответствующие определенной графической сцене, поступают в модуль генерации кода, где на их основе формируется код на языке JavaScript. Приведем некоторую часть программного кода основной функции, сгенерированного на основе правил, описанных выше:

```
function placement()
{
     var graphicalObject = new THREE.Object3D();
     graphicalObject.name = 'Truck_test_1';
     graphicalObject.position.set(-420, -5, 150);
     var material = new THREE.MeshPhongMaterial({side:
     THREE.DoubleSide,color: 0xcc334d, opacity: 0.5,
     shading: THREE.FlatShading, overdraw: 0.5,
     wireframe: 0} );
     var polygonalGeometry = new THREE.Geometry();
     polygonalGeometry.vertices.push(new
     THREE.Vector3(4, 0, -15));
     polygonalGeometry.faces.push(new
     THREE.Face3(0,1,2, new THREE.Vector3(1, 0, 0)));
     polygonalGeometry.faces.push(new
     THREE.Face3(6,7,8, new THREE.Vector3(1, 0, 0)));
     polygonalGeometry.computeFaceNormals();
     var graphicalElement = new
     THREE.Mesh(polygonalGeometry, material);
     graphicalObject.add(graphicalElement);
     graphicalObject.rotation.x += Math.PI/180*0;
     graphicalObject.rotation.y += Math.PI/180*270;
     graphicalObject.rotation.z += Math.PI/180*0;
     scene.add(graphicalObject);
}
```

В данном случае функция placement() содержит код создания объекта «Truck_test_1» и размещения его на сцене. Часть объекта состоит из полигональной сетки, описываемой как набор вершин и граней. Пример графической сцены (рис. 4) содержит несколько подобных объектов, но ввиду идентичности их программного кода, внутри функции описан только один.

6.1 Визуализация

Завершающий этап технологии визуализации – непосредственно отрисовка графической сцены в браузере. Код функций placement(), extendedPlacement() и animation() встраивается в формируемый модулем визуализации код пустой графической сцены, тем самым наполняя ее объектами и событиями. Полученная графическая сцены отображается при помощи браузера и принимает входные данные, определяющие ее поведение, образуя тем самым подсистему визуализации.

Для оценки предложенной технологии визуализации проанализируем производительность более сложной модели, часть которой представлена на рис. 4. Данный пример является частью более сложной сцены, представляющей собой модель промышленной площадки. Сцена разрабатывалась в два этапа: изначально была построена упрощенная сцена, которая в дальнейшем уточнялась и усложнялась за счет детализации.

При детализации модели количество сложных объектов возросло незначительно (с 106 до 119), а количество простых элементов – почти в 6 раз (с 419 до 2453) (рис. 5). При этом возросло время загрузки сцены, а производительность упала с 60 до 20 FPS (frames per second).

Существует исследование, согласно которому минимальное значение частоты кадров, обеспечивающее плавное восприятие человеком отображения движения – 8 FPS [2]. Таким образом, можно считать, что предлагаемая технология обеспечивает достаточную производительность визуализации даже при возрастающей сложности модели.

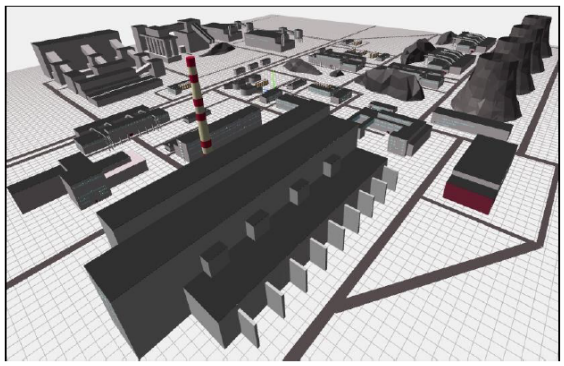

Рис. 5: Графическая модель промышленной зоны.

4. Заключение

В данной работе авторами были предложены основные принципы работы программного средства для визуализации графических сцен, которое способно предоставить пользователям инструментарий для создания подсистем визуализации различного назначения. Была описана архитектура средства, все основные функциональные компоненты средства, а также последовательно разобран процесс создания графической сцены на всех ее основных этапах: создание объектов, формирование правил построения сцены, генерация кода и визуализация.

Базовая функциональность всех основных программных модулей также была реализована авторами.

Предлагаемое программное средство позволит создавать пользовательские подсистемы визуализации, не прибегая к стороннему программному обеспечению и не обладая навыками программирования. При этом авторы отмечают, что такой унифицированный подход к визуализации имеет и отрицательную сторону – создаваемые графические сцены чаще всего имеют более слабую детализа-

цию, чем при использовании профессиональных графических редакторов.

Литература

[1] 3D Warehouse. URL: https://3dwarehouse.sketchup.com.

[2] Keval H., Sasse M.A. To catch a thief – you need at least 8 frames per second: the impact of frame rates on user performance in a CCTV detection task // Proceedings of the 16th ACM international conference on Multimedia. ACM, 2008. P. 941-944

[3] Герасимов С.И., Бутова С.В. Съемка движущегося ракетного поезда // Научная визуализация. 2015. № 2. С. 12-20.

[4] Грибова В.В., Петряева М.В., Федорищев Л.А. Разработка виртуального мира медицинского компьютерного обучающего тренажера // Дистанционное и виртуальное обучение. 2011. №9.

[5] Джамбруно М. Трехмерная (3D) графика и анимация. 2-е издание. пер. с англ. М.: Вильямс, 2002. 640 с. [Giambruo M. 3D Graphics and Animation. 2nd ed. New Riders Press, 2002. 640 p.]

[6] Документация спецификации Web-based Graphics Library (WebGL). 2016. URL: https://www.khronos.org/registry/webgl/specs/latest/1.0/ (дата обращения: 10.03.2016).

[7] Коршунов С.А., Николайчук О.А., Павлов А.И. Web-ориентированный компонент продукционной экспертной системы // Программные продукты и системы. – 2015. №2. С.20-25.

[8] Рыбкина А.И., Бобков А.Е., Никифоров О.В., Пятыгина О.О. Программно-аппаратный комплекс для визуализации геофизических данных на сферическом экране // Научная визуализация. 2015. № 2. С. 38-49.

[9] Рябинин К.В. Методы и средства разработки адаптивных мультиплатформенных систем визуализации научных экспериментов: автореф. дис. . . . канд. физ.-мат. наук: 05.13.11. М., 2015. 23 с.

[10] Трухан И.А., Трухан Д.А. Визуализация учебной информации в обучении математике, ее значение и роль // Успехи современного естествознания. 2013. № 10. С. 113-115.

[11] Турпалов В.Е., Гаврилов Н.И. Технологии трехмерной научной визуализации и геометрического моделирования в цифровой биомедицине // Научная визуализация. 2015. № 4. С. 27-43.