§1 ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Сморкалов А.Ю.

ДИЗАЙН И АРХИТЕКТУРА СРЕДЫ ВЫПОЛНЕНИЯ ЯЗЫКА ПРОГРАММИРОВАНИЯ ВИРТУАЛЬНОЙ РЕАЛЬНОСТИ

Аннотация: *С расширением масштаба применения виртуальных миров в образовании становится необходимым предоставить учителям и ученикам не только возможность размещать в виртуальной реальности объекты, но и задавать поведение и взаимодействие объектов. В данной статье описывается язык программирования vJS, разработанный для виртуального мира vAcademia, а также эффективная программная архитектура среды выполнения этого языка. vJS базируется на стандартном языка JavaScript, язык расширен набором функций, делающим программирование виртуальном реальности максимально простым. vJS позволяет программировать многопользовательские тренажеры, симуляции и игры, организовывать программную поддержку ролевых игр и, так называемых, «серьезных игр». В отличие от аналогов vJS исполняется на каждом клиенте виртуального мира, основан на задании реакции на события 3D-объекта, программа привязана не к 3D-объекту, а к локации, язык содержит большое число функций, результат выполнения которых автоматически синхронизируется. Процесс взаимодействия пользователя с vJS-программой может быть записан в форме 3D-записи, что имеет большое практическое значение.*

Ключевые слова: *виртуальные миры, виртуальная реальность, язык программирования, vJS, JavaScript, тренажеры, симуляции, синхронизация, ролевые игры, серьезные игры*

1. Введение.

Виртуальные миры год от года находят все большее применение в сфере образования. На текущий момент, виртуальные миры применяются для обучения в сотнях университетов по всему миру [1], включая такие престижные учебные заведения как Стэнфорд и Кэмбридж. Виртуальные миры позволяют проводить лекционные и практические занятия, семинары, круглые столы, тренинги, мозговые штурмы.

DOI: 10.7256/2305-6061.2014.1.11328

При цитировании этой статьи сноска на DOI обязательна

Все права принадлежат издательству © NOTA BENE (ООО «НБ-Медиа») www.nbpublish.com

Среди образовательных виртуальных миров одним из самых известных является виртуальный мир vAcademia. vAcademia специально разработан для образования, содержит множество инструментов предназначенных для обучения [2, 3]. Кроме того, каждое занятие в vAcademia может быть записано в виде 3D-записи [4], которая отличается от видеоролика тем, что при просмотре записи учащийся может разместиться в любой точке виртуальной аудитории и в полной мере получить эффект присутствия на занятии.

В vAcademia, как и в большинстве виртуальных миров, пользователи могут разместить свои собственные 3D-модели. Таким образом, становятся доступны простейшие ролевые игры, занятия с демонстрацией моделей сложных механизмов. Однако одно и многопользовательские тренажеры, симуляции и игры на основе простого размещения 3D-объектов реализовать невозможно, т.к. для этих видов образовательной деятельности требуется возможность программирования поведения 3D-объектов. Для решения этой проблемы был разработан язык vJS, который позволяет управлять взаимодействием и поведением объектов в vAcademia.

2. Обзор языков программирования в виртуальных мирах.

LindenLab Scripting Language (LSL) – один из самых известных скриптовых языков, применяемых в виртуальных мирах. Он основан на именованных состояниях [5], в рамках каждого из которых задаются реакции на происходящие в виртуальном мире события. Как только скрипт прикреплен к объекту, он начинает работать. К одному объекту могут быть прикреплено несколько скриптов и все они будут выполняться. Скрипт может изменять как большинство свойств своего объекта, так и взаимодействовать с другими объектами.

Для использования предлагается около 300 встроенных функций, а кроме того пользователи могут объявлять свои функции. LSL – строго типизированный язык, который компилируется в байт-код и исполняется виртуальной машиной на сервере виртуального мира. LSL поддерживает следующие типы данных: целочисленные, с плавающей запятой, строки, 3D-векторы и кватернионы. Не поддерживаются массивы, что является большим недостатком языка. Вместо массивов используют специальные функции, которые возвращают нужное значение по индексу, являющемуся их входным параметром. Кроме того, в LSL не используется объектно-ориентированное программирование и нет средств для повторного использования скриптов [6].

Изначально LSL был разработан как самостоятельный язык с оригинальной виртуальной машиной. Начиная с 2008 года LSL транслируется в язык платформы Microsoft .NET и выполняется с помощью свободной реализации этой платформы Mono. LSL-скрипт исполняется на сервере, поэтому не требуется специальных механизмов синхронизации, однако это повышает нагрузку на сервера виртуального мира и повышает уязвимость серверов, т.к. написанный пользователем код может создавать излишнюю нагрузку на сервер или приводить к повисаниям или падениям.

В виртуальном мире BlueMars используется скриптовый язык Lua [6], расширенный специальным набором функций. Lua – это мощный язык, нашедший широкое применение в игровой индустрии. Он поддерживает динамическую типизацию, основные базовые

Все права принадлежат издательству © NOTA BENE (ООО «НБ-Медиа») www.nbpublish.com

DOI: 10.7256/2305-6061.2014.1.11328
При цитировании этой статьи сноска на DOI обязательна

типы данных. Массивы поддерживают с помощью таблиц, которые по существу являются гетерогенным ассоциативным массивом.

В BlueMars в язык Lua был добавлен набор функций для работы с чатом, лицевой анимацией, динамического создания окон интерфейса и поддержки многопользовательского режима. Поддержка совместной деятельности в языке реализована на низком уровне, т.к. основная цель языка – создание мини-игр. Для создания мини-игр разработан специальный набор функций MultiplayerMinigameAPI, который позволяет достаточно просто разрабатывать простейшие мини-игры на несколько пользователей.

В статье [8] предлагается новый язык Emerson на основе JavaScript, который по утверждению авторов решает три основные проблемы программирования в виртуальных мирах: асинхронный обмен данными между объектами виртуальной среды, вопрос источника корректных данных из нескольких существующих для каждого конкретного случая и поддержка изменения мира в реальном времени с сохранением работоспособности Emerson-программ. Описанные проблемы решаются с помощью трех подходов: создание новых сущностей по шаблону с наследованием уже заданного поведения, изоляция сущностей и возможность задания их независимого поведения, событийная модель программирования с возможностью обработки событий по маске события. Каждая сущность имеет внутри objects, над которыми можно выполнять локальные доверенные синхронные операции, и presenses, через которые доступны недоверенные асинхронные операции

Помимо разделения сущностей на объекты с синхронными и асинхронными операциями, Emerson предлагает использовать различную нотацию для вызова синхронных и асинхронных операций, причем синтаксис асинхронных операций напоминает Erlang и противоречит синтаксису JavaScript [9]. Также сильно отличается от синтаксиса JS и способ задания обработчиков событий с помощью масок событий. Emerson использует более общий механизм обработки событий, однако в наиболее распространенных случаях он оказывается избыточно сложным. В Emerson не предоставляется общего механизма синхронизации виртуального мира, однако предлагается возможность обмена сообщениями между сущностями виртуального мира с возможностью сериализации параметров произвольного типа. Это подразумевает необходимость реализации синхронизации самостоятельно каждым программистом на языке Emerson.

Таким образом, рассмотренные языки не могут быть признаны в полной мере подходящими для программирования виртуальной реальности. Это связано как с несовершенством языковых средств, так и с проблемами при осуществлении синхронизации, которые не решаются на уровне языка программирования. Кроме того, опасным представляется выполнение пользовательских программ на стороне сервера (LSL), что грозит возможными уязвимостями и нарушением стабильности работы серверов виртуального мира. В данной статье описывается язык программирования виртуальной реальности vJS, разработанный для виртуального мира vAcademia, который не имеет описанных недостатков.

Все права принадлежат издательству © NOTA BENE (ООО «НБ-Медиа») www.nbpublish.com

DOI: 10.7256/2305-6061.2014.1.11328

При цитировании этой статьи сноска на DOI обязательна

3. Возможности виртуального мира vAcademia по работе с 3D-объектами.

Как и многие другие виртуальные миры, vAcademia позволяет расставлять заранее подготовленные разработчиками 3D-объекты: виртуальные доски и предметы мебели. Однако, помимо этого, vAcademia позволяет загрузить свою собственную 3D-модель, поменять текстуры и настроить другие свойства. Более того, возможна настройка свойств интерактивности, которые позволяют задать взаимодействие 3D-объекта с посетителями виртуальной среды.

3.1. Подсистема загрузки пользовательских 3D-моделей является основой системы поддержки интерактивных пользовательских объектов. Непосредственно загрузка 3D-модели инициируется скриптовой средой клиента виртуального мира (рис. 1). 3D-модель загружается в виде отдельного файла в формате Collada, 3DS, OBJ, SketchUp или в виде архива с моделью в этих форматах и её текстурами. Движок виртуального мира через систему сетевого взаимодействия направляет запрос на загрузку серверу ресурсов. Сервер ресурсов организует загрузку файла, а после её окончания система постобработки загруженных файлов FilterPalette распознает загруженные 3D-модели и передает их на обработку классу ModelConverter, который распаковывает архив с 3D-моделью и отдает директорию с распакованной моделью на обработку конвертору 3D-моделей.

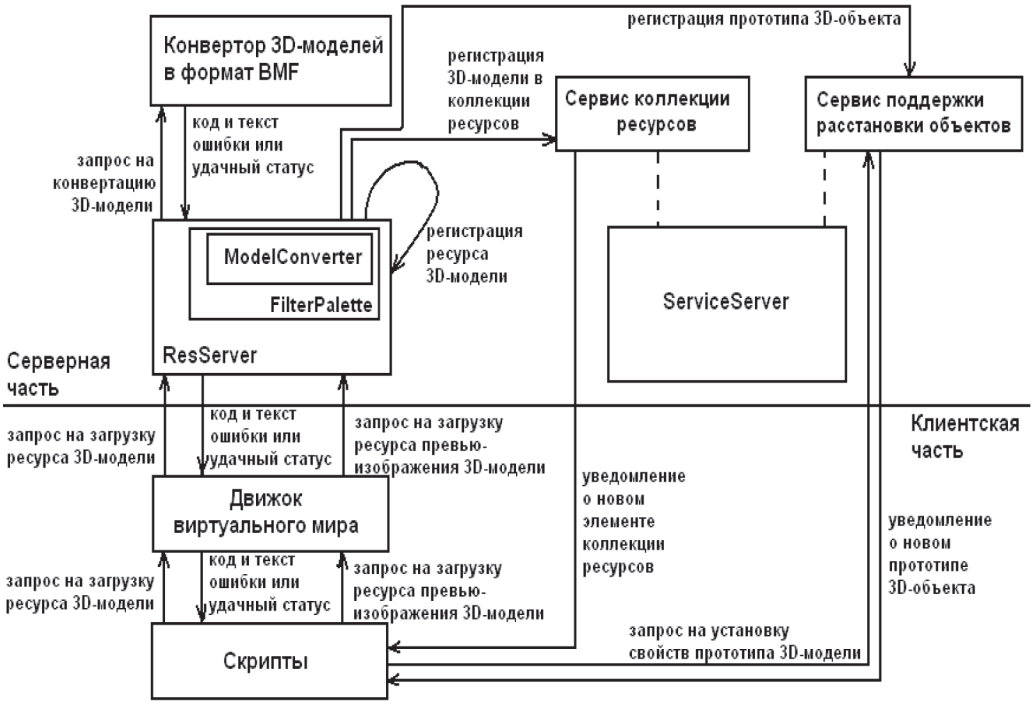

Рис. 1. Схема загрузки пользовательской 3D-модели

Конвертер 3D-моделей представляет собой отдельное приложение, которое с помощью библиотеки Assimp [10] преобразует 3D-модель в формат BMF5, а текстуры в изображения форматов PNG и JPEG разумного разрешения (не более 1024 на 1024

Все права принадлежат издательству © NOTA BENE (ООО «НБ-Медиа») www.nbpublish.com

DOI: 10.7256/2305-6061.2014.1.11328
При цитировании этой статьи сноска на DOI обязательна

пикселей) с линейными размерами кратными степени двойки, что необходимо для отображения на устаревших моделях видеокарт. Перечисленные форматы поддерживаются vAcademia без необходимости дополнительных преобразований. Конвертер 3D-моделей возвращает код и текст ошибки или статус корректного завершения (в этом случае генерируется сконвертированная модель). Конвертер также проверяет соответствие 3D-модели ограничениям на число полигонов и количество текстур. Код и текст ошибки возвращаются по цепочке клиент-серверного взаимодействия скриптовой среде.В случае удачной конвертации ModelConverter регистрирует сконвертированную модель в сервере ресурсов, чтобы она могла быть загружена клиентом виртуального мира, регистрирует 3D-модель в сервисе коллекции ресурсов (чтобы она была доступна в коллекции ресурсов в папке «Мои 3D-модели»), а также регистрирует прототип 3D-объекта в сервисе поддержки расстановки объектов (чтобы он был доступен в списке объектов галереи инструмента расстановки объектов). Перечисленные сервисы уведомляют скриптовую среду о произошедших изменениях.

После успешной загрузки модели скриптовая среда генерирует превью-изображение новой 3D-модели и запрашивает у пользователя первоначальные размеры и свойства коллизий. Превью-изображение загружается на сервер-ресурсов, а свойства объектов отправляются запросом на сервис поддержки расстановки объектов для внесения изменений в прототип.

3.2. Подсистема расстановки пользовательских 3D-объектов. Пользовательские объекты могут быть установлены в назначенном занятии или временной локации. Такие ограничения связаны с тем, что пользовательские 3D-модели обычно неоптимизированны с точки зрения количества полигонов и их одновременное использования в нескольких локациях основной реальности может привести к значительному снижению скорости работы клиентов виртуального мира других посетителей виртуального мира. Назначенное занятие проходит в отдельной реальности, поэтому использование объектов повлияет только на комфорт работы участников занятия. Объекты во временной локации удаляются сразу после окончания занятия, а их дальность видимости снижена вдвое. Эти меры позволяют реализовать компромиссный вариант организации занятий с пользовательскими объектами в основной реальности.

Пользовательские объекты всегда создаются из прототипа, который хранится на сервисе поддержки размещения объектов. Прототип содержит в себе путь к ресурсу 3D-модели на сервере ресурсов, начальный поворот, настройки коллизий и другие свойства. После первоначального размещения объект может быть выделен, перемещен, повернут, у него можно изменить масштаб, его можно клонировать и удалить.

3.3. Подсистема настройки свойств пользовательских 3D-объектов. Одной из важнейших возможностей системы расстановки объектов является настройка свойств. Большая часть свойств применяется не только к текущему выбранному объекту, но и к его прототипу. Однако, если в прототип вносятся изменения, они не распространяются на все ранее расставленные объекты и будут применяться только к вновь создаваемым. У пользовательского 3D-объекта могут быть настроены общие свойства: отображение

Все права принадлежат издательству © NOTA BENE (ООО «НБ-Медиа») www.nbpublish.com

DOI: 10.7256/2305-6061.2014.1.11328

При цитировании этой статьи сноска на DOI обязательна

теней, тип коллизии, запреты для других пользователей на изменение свойств, перемещение, поворот, масштабирование и удаление.

Также можно настроить размещение сидячих мест на пользовательском объекте, поменять текстуру, настроить автовращение вокруг произвольной оси с произвольной скоростью, привязать к объекту воспроизведение 3д-звука с задержкой между повторами. Эти свойства не задают взаимодействие с пользователем, однако позволяют удобно собрать некоторую 3D-среду из готовых объектов.

Свойства интерактивности позволяют инициировать в случае клика по объекту одно или несколько (если они не противоречат друг другу) из следующих действий:

- Вывод текстового сообщения (только тому, кто кликнул)
- Воспроизведение звука (для всех в текущей локации)
- Перемещение в точку (только того, кто кликнул)
- Перемещение в просмотр записи (только того, кто кликнул)
- Переход на сайт (только у того, кто кликнул)
- Смена текстуры на объекте на заданную(для всех в текущей локации)
- Вращение (для всех в текущей локации)
- Подойти по клику (только для того, кто кликнул)

Общие свойства и свойства интерактивности реализованы с помощью скриптовых синхронизируемых свойств и обработки их изменения в скрипте (для выделенного объекта, применяются на всех компьютерах посетителей виртуального мира) и свойств прототипа в JSON-нотации [11], хранимых в MySQL-базе данных сервиса поддержки размещения объектов и передаваемых на клиент виртуального мира по запросу.

3.4. Подсистема поддержки шаблонов локаций. Шаблон локации представляет собой список всех размещенных в локации пользовательских объектов со всеми их свойствами. Шаблон локации может быть в определенный момент сохранен, а затем при необходимости применен (объекты из шаблона будут вновь созданы и расставлены в соответствии с записанными свойствами). Это удобно для быстрого восстановления заранее подготовленной 3D-сцены для проведения занятия.

При создании шаблона происходит сериализация списка объектов с их свойствами в JSON-объект. Одновременно автоматически генерируется 2D превью-изображение шаблона локаций, которое в виде изображения загружается на сервер ресурсов. Сериализованные данные, путь к превью и название сохраняются в качестве шаблона с помощью запроса на сервис поддержки размещения объектов.

При применении шаблона с сервиса поддержки размещения объектов запрашивается сериализованный список объектов. После получение списка происходит десериализация, синхронизированное создание объектов по списку и применение их свойств.

3.5. Локация «Мой дом» - это специальный участок 3D-пространства, который уникален для каждого пользователя и доступен для заполнения пользовательскими объектами. Его содержимое не меняется со временем иначе как по желанию владельца локации, что позволяет обустроить личное пространство или собственную аудиторию.

Мой дом представляет собой отдельный остров, удаленный на значительное рас-

Все права принадлежат издательству © NOTA BENE (ООО «НБ-Медиа») www.nbpublish.com

DOI: 10.7256/2305-6061.2014.1.11328
При цитировании этой статьи сноска на DOI обязательна

стояние от других островов vAcademia. Уникальность локации для пользователя технически обеспечивается с помощью выделения «Моего дома» каждого пользователя в отдельную независимую реальность, что позволяет для каждого пользователя хранить свой набор синхронизируемых объектов с соответствующими синхронизируемыми свойствами. Отдельный остров и уникальность реальности позволяют значительно смягчить ограничения на общее число полигонов расставленных пользовательских объектов, т.к. отображаемые объекты не могут помешать комфортной работе других пользователей, а сложность отображения пустого «Моего дома» значительно меньше, чем отображения главного острова vAcademia.

3.6. Применение. Интерактивные пользовательские объекты применяются в настоящий момент для занятия по изучению иностранных языков, проведения ситуационных тренингов, демонстрации моделей сложных устройств и молекул, а также для представления ссылок на учебные материалы в 3D-пространстве (рис. 2). Их применение позволяет сделать занятия более интересными, создать эффект присутствия в некоторой проблемной ситуации или предоставить возможность всестороннее ознакомиться с моделями сложных механизмов.

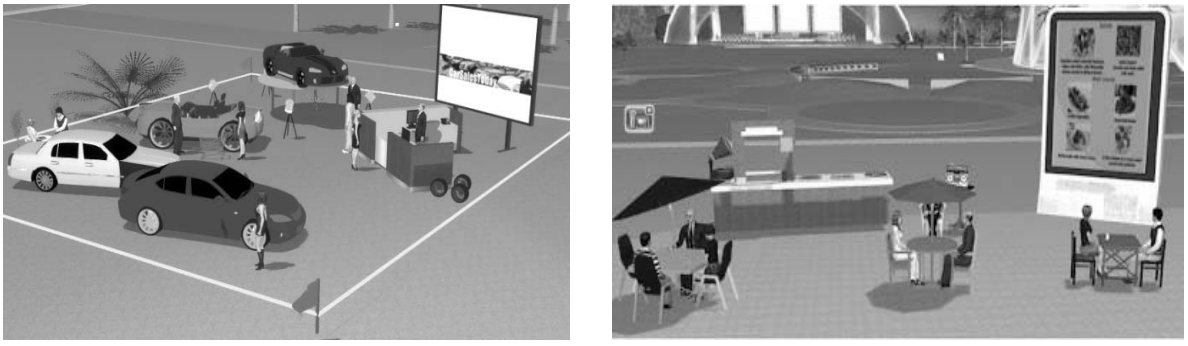

Рис. 2. а) Тренинг по продаже автомобилей
б) Урок английского языка в декорациях летнего кафе

Поддержка форматов Collada и SketchUp позволяет использовать огромную библиотеку бесплатных 3D-моделей Google 3dwarehouse [12], а другие поддерживаемые форматы, 3DS и OBJ, считаются одними из самых популярных форматов в сети интернет. При необходимости использования вручную сделанной модели, возможен экспорт из 3D-редактора в один из поддерживаемых форматов.

4. Дизайн языка программирования vJS

4.1. Основные идеи в основе vJS

При разработке дизайна языка программирования vJS особое внимание уделялось предоставлению максимально простых абстракций, позволяющих реализовывать задачи из предметной области виртуальной реальности минимальными усилиями. vJS основывается на языке JavaScript, все стандартные функции и типы данных этого языка доступны

Все права принадлежат издательству © NOTA BENE (ООО «НБ-Медиа») www.nbpublish.com

DOI: 10.7256/2305-6061.2014.1.11328

При цитировании этой статьи сноска на DOI обязательна

для использования, кроме того синтаксис языка не расширялся и не изменялся. Это позволило избежать проблем LSL и Lua при работе с массивами, а также дало возможность использовать vJS-программистами знакомый синтаксис языка в отличие от Emerson.

Любое занятие, в том числе с использованием тренажера или симуляции, может происходить только внутри одной локации. Поэтому язык позволяет управлять всеми пользовательскими объектами внутри одной локации, обращаясь к ним по имени. Имя объекта разработчик vJS-программы задает самостоятельно через графический интерфейс vAcademia. По имени объекта можно получить ссылку на него и, обращаясь по ссылке, переопределить обработчики стандартных событий 3D-объекта. Размещение объектов происходит визуально с помощью инструмента размещения объектов. Программное создание новых объектов не поддерживается, т.к. представляет значительную сложность для vJS-программиста ввиду необходимости задания в программном коде многочисленных свойств 3D-объекта.

Надо отметить, что программа на vJS не имеет смысла без объектов, которыми она управляет. Поэтому предусмотрено, что работа программы не начинается до тех пор, пока все объекты, используемые программой, не будут загружены. Только после окончания загрузки используемых объектов программа компилируется и запускается, а после запуска тот факт, что объекты загружены, гарантирует получение корректных ссылок на все используемые объекты внутри программы.

Типовая программа на vJS состоит из следующих шагов:

1. Получить ссылки на управляемые объекты
2. Назначить обработчики необходимых событий для этих объектов

Таким образом, работа vJS-программы основывается на принципах событийного программирования. Примером программы на vJS может программа из листинга 1.

```
var obj1 = scene.getObjectByName(«ИмяМоегоОбъекта»);
obj1.onPress = function()
{
// сделать что-нибудь, если нажали на объект
}
```

Листинг 1. Код простейшей программы на vJS

Ссылка на объект получается с помощью вызова метода getObjectByName глобального объекта scene. Обычно единственный параметр этой функции является фиксированно заданной строкой. Зачастую оказывается удобным использовать обращения к объектам по именам из строковых переменных. Такой вариант предусмотрен при разработке дизайна языка: требуется во избежание некорректной работы вашей программы в начале программы разместить вызовы метода requireObject глобального объекта scene. Выполнение программы также не начнется, пока объекты, имена которых будут перечислены в вызовах requireObject, не будут загружены.

Все права принадлежат издательству © NOTA BENE (ООО «НБ-Медиа») www.nbpublish.com

DOI: 10.7256/2305-6061.2014.1.11328
При цитировании этой статьи сноска на DOI обязательна

4.2. Событийная модель программирования в vJS

У 3D-объекта предусмотрено несколько стандартных событий, на которые в vJS-программе могут быть заданы обработчики:

- onPress – нажата кнопка мыши на объекте
- onRelease – кнопка мыши отпущена на объекте
- onReleaseOutside – кнопка мыши отпущена вне объекта
- onRollOver– указатель мыши попал на объект
- onRollOut – указатель мыши вышел из объект
- onKeyDown– нажата клавиша клавиатуры
- onKeyUp – отпущена клавиша клавиатуры

Перехват клавиатурных событий является проблемным местом при программировании в виртуальной реальности, т.к. в отличие от 2D-программ отсутствует элемент в фокусе, который получает такие события. Поэтому был предусмотрен такой гибкий подход, как подписка на клавиатурные события, который определяет, какие объекты будут получать клавиатурные события в данный момент времени.

Чтобы события нажатия клавиш приходили определенному 3D-объекту, необходимо подписать его на получений событий от клавиатуры принудительно с помощью метода addListener глобального объекта Key с параметром-ссылкой на 3D-объект. Чтобы отписать 3D-объект от клавиатурных событий необходимого использовать метод removeListener.

4.3. Управление свойствами 3D-объектов в vJS

vJS-программа управляет 3D-объектами, поэтому в языке необходимо предусмотреть управление основными свойствами 3D-объектов, а именно: позицией, поворотом и масштабом. У каждого 3D-объекта предусмотрены стандартные свойства position, rotationи scale, для которых используются соответствующие типы данных Position3D, Rotation3D, Scale3D. Эти типы данных позволяют как обращаться к свойствам 3D-объекта покомпонентно, так и вызывать специфические для каждого типа данных методы, такие как например, скалярное произведение векторов или вычисление обратного поворота.

Надо отметить, что Rotation3D представляет собой не обертку над кватернионом (как в языке LSL), который является слишком сложной сущностью для прикладного программиста, а новый тип данных. Поворот задается осью вращения и углом поворота, измеряемым в градусах. Конвертация заданного таким образом поворота в кватернион, используемый в 3D-графике, происходит внутри системы исполнения скриптов без участия vJS-программиста.

4.4. Стандартные несинхронизируемые функции в vJS

Неотъемлимой частью трехмерных тренажеров и симуляций является организация анимации, повторение некоторых действий через промежутки времени. С целью поддержки такого рода вычислений была введена глобальная функция playByTimer. Первым

Все права принадлежат издательству © NOTA BENE (ООО «НБ-Медиа») www.nbpublish.com

DOI: 10.7256/2305-6061.2014.1.11328

При цитировании этой статьи сноска на DOI обязательна

параметром она принимает пользовательскую функцию глобального объекта scene, а вторым параметром задержку в миллисекундах, через которую эта функция должна однократно сработать. Если функция должна вызываться периодически, она может ставить на исполнение по таймеру сама себя.

Виртуальная реальность предполагает многопользовательскую среду, поэтому при программировании часть действий должны выполняться для всех пользователей, а другая часть действий должны выполняться только для одного пользователя в зависимости от постановки задачи. Для выполнения локальных операций существуют встроенные функции, которые выполняются только на одном клиенте виртуального мира (на том, который инициировал их исполнение). В числе таких глобальных функций:

- showMessageBox – отображает окошко с текстом и кнопку OK для его закрытия. Первый параметр – текст в заголовке окна, второй – текст в самом окне.
- showConfirmation – отображает окошко с текстом и кнопки «Да»/ «Нет». Первый параметр – текст в заголовке окна, второй – текст в самом окне. Функция возвращает объект, у которого необходимо реализовать функции onYes, onNo, onClose для обработки возможных вариантов действий пользователя.
- showInputBox – отображает окно с заголовком, вопросом, полем ввода, кнопками «Да» и «Нет». Функция возвращает объект, у которого необходимо реализовать функции onYes, onNo, onCancel обработки возможных вариантов действий пользователя. Внутри обработчика onYes получить введенное значение можно с помощью метода getInput.
- openURL – открывает окошко браузера с необходимым интернет-адресом. Адрес задается первым параметром функции.

Таким образом, несинхронизируемые функции предназначены для организации диалога с отдельным пользователем, предоставлении ему дополнительных сведений.

4.5. Стандартные автоматически синхронизируемые функции и свойства в vJS

Вторая часть стандартных функций является синхронизируемой, т.е. результат их действия виден для всех посетителей виртуального мира. Дизайн языка предусматривает автоматическую синхронизацию, т.е. синхронизируемая функция должна быть однократно вызвана, после чего результат её выполнения синхронизируется автоматически без вмешательства vJS-программиста. В числе таких функций:

- teleport – глобальная функция для переноса аватара своего пользователя в определенную точку мира. Точка мира задается тремя параметрами - числами-координатами в порядке X, Y, Z. Функция не переносит аватаров всех пользователей, однако перенос одного аватара виден для всех.
- setVisible – метод каждого 3D-объекта для управления его видимостью.

Надо отметить, что только простейшие синхронизируемые методы могут использовать параметры базовых типов данных. Для более сложных синхронизируемых действий

Все права принадлежат издательству © NOTA BENE (ООО «НБ-Медиа») www.nbpublish.com

DOI: 10.7256/2305-6061.2014.1.11328
При цитировании этой статьи сноска на DOI обязательна

необходимо использовать серверные ресурсы, например, звуки или изображения, хранимые на сервере. Т.к. vJS работает в многопользовательской среде использование локальных ресурсов (хранимых на компьютере одного пользователя) не имеет смысла.

Для того, чтобы получить ссылку на необходимый серверный ресурс, предусмотрена глобальная функция getServerResource, параметром которой является название файла в коллекции ресурсов. Ссылка должна быть сохранена в глобальный объект scene для дальнейшего использования и должна быть получена в начале программы для организации эффективного клиент-серверного взаимодействия, возможности предварительно закешировать ресурс до непосредственного его первого использования. Кеширование и загрузка серверного ресурса на клиент организуются автоматически без участия vJS-программиста.

В языке vJS предусмотрены следующие синхронизируемые функции, использующие серверные ресурсы:

- playMusic – метод 3D-объекта, первым параметром указывается серверный ресурс со звуком (mp3 или wav-файл), а вторым параметром громкость звука в процентах.
- replaceTexture – метод 3D-объекта, первый параметр - номер материала в 3D-объекте для замены (номера начинаются с 0), второй параметр – серверный ресурс с текстурой (JPEG или PNG).

Помимо синхронизируемых функций предусмотрены стандартные синхронизируемые свойства 3D-объекта: position, rotation и scale. Их изменения синхронизируются автоматически.

5. Архитектура среды выполнения vJS

Внутренний скриптовый язык, на котором написана примерно треть всего программного кода vAcademia, представляет собой смесь языка JavaScript и XML для декларативного задания объектов, их свойств, обработчиков событий, скриптов инициализации. Скрипты, написанные на этом языке, выполняются на стороне клиента виртуального мира. Т.к. внутренний язык уже использует JavaScript, было принято решение о разработке языка vJS как ограниченного подмножества внутреннего скриптового языка. Таким образом, для того, чтобы реализовать стандартные функции языка vJS и методы 3D-объектов языка vJS, потребовалось завести соответствующие глобальные функции и методы в классе пользовательских 3D-объектов.

Такой подход может иметь недостаток в виде теоретической возможности получения доступа из vJS-программы к возможностям всего языка с целью взлома системы и несанкционированного доступа к каким-либо ресурсам или услугам. Однако, доступ ко всем защищенным ресурсам предоставляется сервером виртуального мира после проверки прав доступа. Таким образом, несанкционированный доступ из vJS-скрипта исключен.

Программа на языке vJS выполняется на каждом клиенте виртуального мира. Для реализации этого в каждой локации автоматически создается синхронизируемый объект поддержки vJS. Синхронизируемый объект поддержки vJS используется для сохранения

Все права принадлежат издательству © NOTA BENE (ООО «НБ-Медиа») www.nbpublish.com

DOI: 10.7256/2305-6061.2014.1.11328

При цитировании этой статьи сноска на DOI обязательна

текста vJS-программы, синхронизации текста между всеми клиентами виртуального мира и восстановления текста программы при заходе в мир. Для реализации синхронизации используется встроенный в vAcademia механизм синхронизируемых свойств.

После того, как новая vJS-программа обновлена через синхронизируемый объект поддержки vJS, начинается процесс запуска vJS-программы по следующему алгоритму:

1. Обработка текста vJS-программы с целью получения списка всех используемых 3D-объектов.
2. Ожидание, когда все используемые в vJS-программе объекты будут загружены.
3. Удаление всех обработчиков событий у объектов из списка используемых vJS-программой с целью приведения объектов в начальное состояние.
4. Запуск vJS-программы с помощью выполнения её текста через метод eval.

Отдельного внимания заслуживает механизм отслеживания загрузки объектов и получения ссылок на них. Каждый пользовательский 3D-объект хранит в себе имя для использования в vJS в виде синхронизируемого свойства. После загрузки 3D-объекта, которое включает в себя создание скриптового объекта и окончание загрузки полигонального представления 3D-модели, происходит установка значений синхронизируемых свойств. В этот момент, при установке свойства имени объекта, происходит регистрация 3D-объекта в системе времени исполнения языка vJS.

Для проверки, загружен 3D-объект или нет, достаточно проверить его наличие в списке объектов системы времени выполнения языка. При удалении 3D-объекта происходит обратная процедура: снятие с регистрации 3D-объекта. Смена имени в режиме реального времени обрабатывается как снятие с регистрации 3D-объекта с последующей повторной регистрацией с новым именем.

Для использования серверных ресурсов был разработан класс серверного ресурса. При вызове глобальной функции getServerResource автоматически создается экземпляр класса серверного ресурса. В этот момент автоматически отправляется запрос на сервис коллекции ресурсов (который осуществляет поддержку пользовательских ресурсов в сервере сервисов vAcademia) для получения серверного адреса ресурса. После получения серверного адреса, он запоминается как свойство экземпляра класса серверного ресурса и начинается предзагрузка ресурса, чтобы в момент необходимости его применения он с большой долей вероятности находился в кеше клиента виртуального мира.

При необходимости применить серверный ресурс, экземпляр класса серверного ресурса возвращает серверный путь ресурса и обращение к нему разрешается стандартным образом ресурсной системой vAcademia. Если ресурс уже был загружен в кеш клиента vAcademia, он используется из кеша, если нет – загружается с сервера. Подобная схема позволяет скрыть от vJS-программиста сложность работы с серверными ресурсами при поддержке наиболее оптимальной схемы работы с ними (предзагрузка).

Синхронизируемые функции языка vJS были реализованы за счет введения дополнительных синхронизируемых свойств, которые соответствуют изменениям вносимым функциями. Например, в случае реализации синхронизируемой функции замены текстур было добавлено синхронизируемое свойство списка замененных текстур.

Все права принадлежат издательству © NOTA BENE (ООО «НБ-Медиа») www.nbpublish.com

DOI: 10.7256/2305-6061.2014.1.11328
При цитировании этой статьи сноска на DOI обязательна

Обработка изменений синхронизируемых свойств позволила достичь визуального совпадения производимых из vJS изменений на всех клиентах виртуального мира. Кроме того, использование механизма синхронизируемых свойств позволяет организовать 3D-запись [13] взаимодействия пользователя с vJS-программой (например, прохождения тренажера), что имеет большое практическое значение.

Программа на языке vJS не имеют смысла без объектов, которыми она управляет. Поэтому для её сохранения для последующего повторного использования был доработан механизм шаблона локаций с тем, чтобы сервис поддержки расстановки объектов сохранял в отдельную таблицу базы данных соответствующий локации текст vJS-программы. При восстановлении локации из шаблона восстанавливается и текст vJS-программы.

Таким образом, была разработана архитектура среды выполнения языка vJS, которая обеспечила корректное исполнение vJS-программы на всех клиентах виртуального мира вне зависимости от прогресса загрузки 3D-объектов виртуального мира. Также была реализована эффективная схема работы с серверными ресурсами, которая позволяет сократить задержки клиент-серверного взаимодействия, упростить и унифицировать работу vJS-программиста по использованию серверных ресурсов.

6. Результаты.

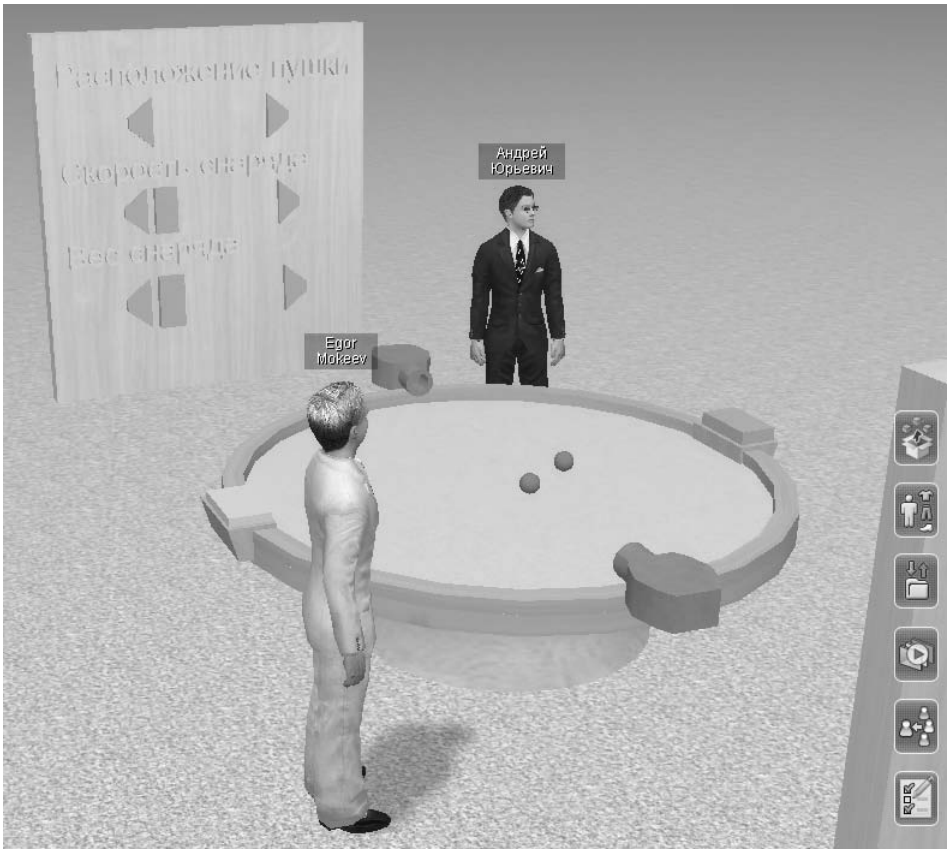

Рис. 3. Тренажер «Столкновение двух тел. Закон сохранения импульса»

Все права принадлежат издательству © NOTA BENE (ООО «НБ-Медиа») www.nbpublish.com

DOI: 10.7256/2305-6061.2014.1.11328

При цитировании этой статьи сноска на DOI обязательна

Разработанный язык был интегрирован в виртуальный образовательный мир vAcademia. В ходе первого семестра 2013-2014 учебного года прошла апробация языка. Язык vJS применялся для выполнения двух лабораторных работ и одного курсового проекта каждым из студентов 5 курса специальности 230105.65 «Программное обеспечение вычислительной техники и автоматизированных систем» Поволжского Государственного Технологического Университета.

Студентами были успешно выполнены поставленные задачи, в том числе разработано несколько курсовых проектов (многопользовательских тренажеров, симуляций, игр), имеющих практическое значение, например:

- Тренажер «Столкновение двух тел. Закон сохранения импульса» (рис. 3)
- Симуляция «Стрельба из лука. Физически точный полет стрелы»
- Игра «Шашки» для двух пользователей (рис. 4)

В ходе апробации студентами был отмечен ряд недостатков в интерфейсе виртуального мира в части поддержки программирования, недостаток средств отладки и невозможность создания и управления ботами. Большая часть этих недостатков на данный момент уже исправлена, а расширение языка vJS для управления ботами находится в стадии активной разработки.

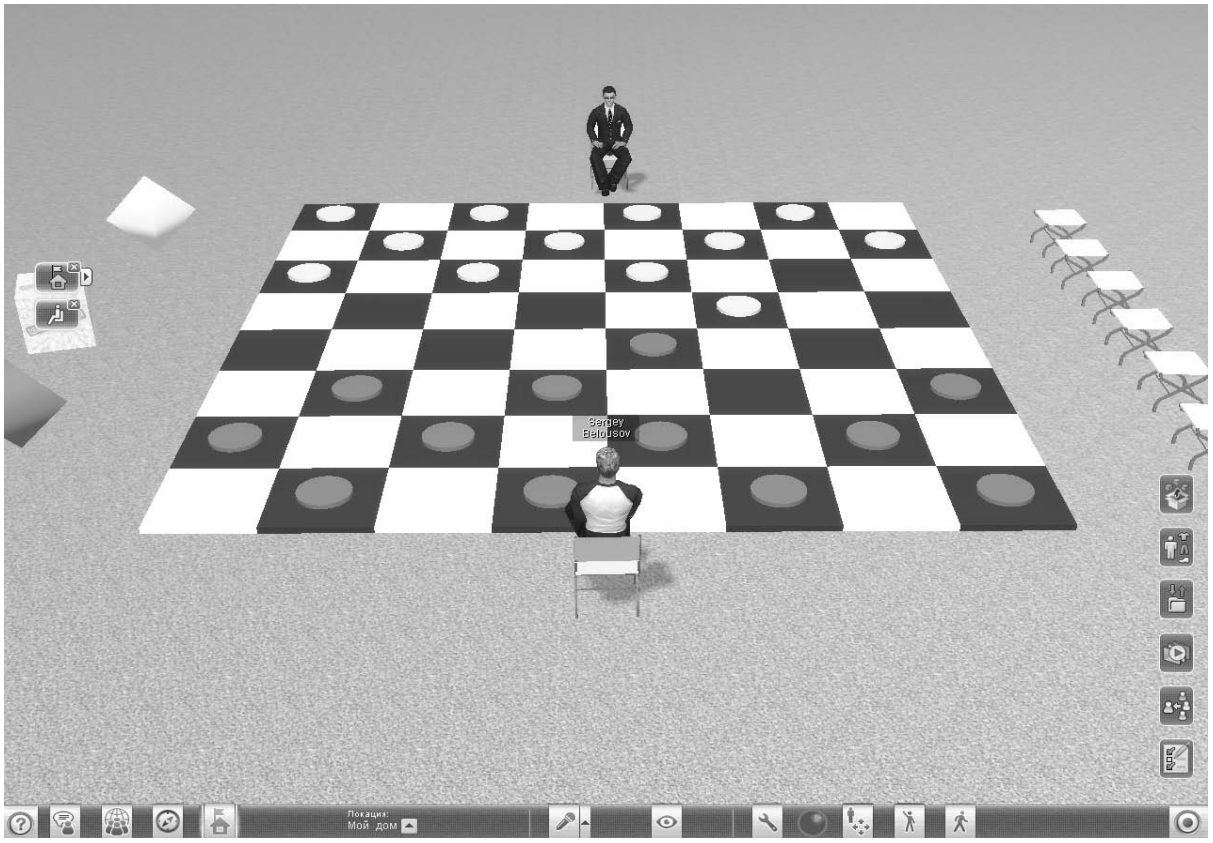

Рис. 4. Игра «Шашки» для двух пользователей

Все права принадлежат издательству © NOTA BENE (ООО «НБ-Медиа») www.nbpublish.com

DOI: 10.7256/2305-6061.2014.1.11328
При цитировании этой статьи сноска на DOI обязательна

6. Выводы

Был разработан язык программирования виртуальной реальности vJS. Язык предназначается для разработки пользователями виртуального мира vAcademia многопользовательских тренажеров, симуляций и игр. В отличие от рассмотренных аналогов язык vJS решает проблему синхронизации на уровне языка, безопасно исполняется на уровне клиента виртуального мира, обладает мощным и простым синтаксисом, обеспечивает простую и эффективную работу с серверными пользовательскими ресурсами. Процесс взаимодействия с vJS-программой может быть записан в форме 3D-записи, что имеет большое педагогическое значение.

Библиография :

1. А. Ю. Сморкалов Реализация образовательных инструментов в виртуальных 3D-средах с использованием потоковых процессоров. // Международный электронный журнал "Образовательные технологии и общество (Educational Technology & Society)" – 2011.-V. 14.-№ 3.-С. 409-425 .-ISSN 1436-4522. URL: http://ifets.ieee.org/russian/periodical/journal.html
2. А.Ю. Сморкалов. Математическая и программная модели генерации текстур на графических потоковых процессорах. // Программные системы и вычислительные методы.-2013.-№ 1.-С. 116-128.
3. A Smorkalov, M Fominykh, M Morozov. Stream Processors Texture Generation Model for 3D Virtual Worlds: Learning Tools in vAcademia. In Proceeding of IEEE International Symposium on Multimedia (ISM), 2013, pp. 17-24
4. М.Е. Рыженков. Редактирование трехмерного образовательного контента. // Программные системы и вычислительные методы.-2013.-№ 1.-С. 95-105.
5. Robert J. Cox, Patricia S. Crowther. A review of Linden Scripting Language and its role in Second Life. In ICCMSN'08 Proceedings of the First international conference on Computer-Mediated Social Networking, pp. 35-47, Springer-Verlag Berlin, Heidelberg, 2009.
6. Cristina Lopes. The Worst Language Ever Designed: The Case for Better Programming Languages for 3D Environments. Report on POPL OBT 2014. URL: http://popl-obt-2014.cs.brown.edu/papers/worst.pdf
7. LUA specification. URL: http://www.lua.org/doc/hopl.pdf
8. B. Chandra, E. Cheslack-Postava, B. F. T. Mistree, P. Levis, D. Gay. Emerson: Scripting for Federated Virtual Worlds. Proc. CGAMES'10, 2010.
9. Behram F.T. Mistree, Bhupesh Chandra, Ewen Cheslack-Postava, Philip Levis, David Gay. In ONWARD'11 Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reflections on programming and software, pp. 77-90, New York, USA, 2011. DOI: 10.1145/2048237.2048247
10. Open Asset Import Library (Assimp). URL: http://assimp.sourceforge.net
11. The application/json Media Type for JavaScript Object Notation (JSON). RFC 4627 Specification. URL: http://tools.ietf.org/html/rfc4627
12. Google 3dwarehouse.URL: http://sketchup.google.com/3dwarehouse/

Все права принадлежат издательству © NOTA BENE (ООО «НБ-Медиа») www.nbpublish.com

DOI: 10.7256/2305-6061.2014.1.11328

При цитировании этой статьи сноска на DOI обязательна

13. Mikhail Morozov, Alexey Gerasimov, Mikhail Fominykh, and Andrey Smorkalov: «Asynchronous Immersive Classes in a 3D Virtual World: Extended Description of vAcademia,» in Marina Gavrilova, Chih Jeng Kenneth Tan and Arjan Kuijper Eds., Transactions on Computational Science (TCS), LNCS 7848, Issue XVI, 2013, Springer.

References:

1. Smorkalov A.Yu. Realizatsiya obrazovatel'nykh instrumentov v virtual'nykh 3D-sredakh s ispol'zovaniem potokovykh protsessorov. // Mezhdunarodnyi elektronnyi zhurnal "Obrazovatel'nye tekhnologii i obshchestvo (Educational Technology & Society)" – 2011.-V. 14.-№ 3.-S. 409-425 .-ISSN 1436-4522. URL: http://ifets.ieee.org/russian/periodical/journal.html
2. A.Yu. Smorkalov. Matematicheskaya i programmnaya modeli generatsii tekstur na graficheskikh potokovykh protsessorakh. // Programmnye sistemy i vychislitel'nye metody.-2013.-№ 1.-C. 116-128.
3. A Smorkalov, M Fominykh, M Morozov. Stream Processors Texture Generation Model for 3D Virtual Worlds: Learning Tools in vAcademia. In Proceeding of IEEE International Symposium on Multimedia (ISM), 2013, pp. 17-24
4. M.E. Ryzhenkov. Redaktirovanie trekhmernogo obrazovatel'nogo kontenta. // Programmnye sistemy i vychislitel'nye metody.-2013.-№ 1.-C. 95-105.
5. Robert J. Cox, Patricia S. Crowther. A review of Linden Scripting Language and its role in Second Life. In ICCMSN'08 Proceedings of the First international conference on Computer-Mediated Social Networking, pp. 35-47, Springer-Verlag Berlin, Heidelberg, 2009.
6. Cristina Lopes. The Worst Language Ever Designed: The Case for Better Programming Languages for 3D Environments. Report on POPL OBT 2014. URL: http://popl-obt-2014.cs.brown.edu/papers/worst.pdf
7. LUA specification. URL: http://www.lua.org/doc/hopl.pdf
8. B. Chandra, E. Cheslack-Postava, B. F. T. Mistree, P. Levis, D. Gay. Emerson: Scripting for Federated Virtual Worlds. Proc. CGAMES'10, 2010.
9. ehram F.T. Mistree, Bhupesh Chandra, Ewen Cheslack-Postava, Philip Levis, David Gay. In ONWARD'11 Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reflections on programming and software, pp. 77-90, New York, USA, 2011. DOI: 10.1145/2048237.2048247
10. Open Asset Import Library (Assimp). URL: http://assimp.sourceforge.net
11. he application/json Media Type for JavaScript Object Notation (JSON). RFC 4627 Specification. URL: http://tools.ietf.org/html/rfc4627.
12. Google 3dwarehouse.URL: http://sketchup.google.com/3dwarehouse/
13. Mikhail Morozov, Alexey Gerasimov, Mikhail Fominykh, and Andrey Smorkalov: "Asynchronous Immersive Classes in a 3D Virtual World: Extended Description of vAcademia," in Marina Gavrilova, Chih Jeng Kenneth Tan and Arjan Kuijper Eds., Transactions on Computational Science (TCS), LNCS 7848, Issue XVI, 2013, Springer.

Все права принадлежат издательству © NOTA BENE (ООО «НБ-Медиа») www.nbpublish.com

DOI: 10.7256/2305-6061.2014.1.11328
При цитировании этой статьи сноска на DOI обязательна