Оптимизация сетевого трафика сообщений синхронизации объектов образовательного виртуального мира vAcademia

Дмитрий Анатольевич Быстров
старший преподаватель кафедры информатики и системного программирования,
Марийский государственный технический университет,
пл. Ленина, 3, г. Йошкар-Ола, 424000, (8362) 68-60-90
uncle.demian@gmail.com

Анотация

В настоящей статье представлен подход синхронизации объектов на основе их прикладных состояний, используемый в образовательном виртуальном мире vAcademia. В настоящие время виртуальные миры широко используются в сфере образования. Актуальной проблемой в виртуальных мирах является обеспечение осведомленности пользователей о действиях других пользователей, находящихся в одном и том же месте виртуального пространства. Эта задача решается путем синхронизации реплик синхронизируемых объектов виртуального мира на компьютерах пользователей. Проведенный анализ подходов синхронизации объектов виртуальных миров показывает, что для обеспечения эквивалентности реплик синхронизируемых объектов на компьютерах разных пользователей необходима большая пропускная способность их Интернет-соединения. Применение представленного подхода позволяет значительно уменьшить сетевой трафик сообщений.

This paper presents an approach object synchronization based on applied object state used in the educational virtual world vAcademia. Today virtual worlds are widely used in eLearning. Topical issue is to ensure users' awareness of whom they interact with and how they interact within a virtual world. This problem decided by synchronizing replicas of virtual world synchronized objects in computers of end users. The analysis approaches to the synchronization objects of virtual worlds has shown that to ensure the equivalence of synchronized object replicas need more bandwidth of Internet connection for each of users. Using the presented approach can significantly reduce network traffic synchronization messages objects of the virtual world.

Ключевые слова

виртуальный мир, синхронизация объектов, осведомлённость
virtual world, object synchronization, user awareness

Введение

Виртуальные миры являются одной из наиболее перспективных сфер в индустрии программного обеспечения. Родившись как одно из направлений индустрии компьютерных игр, в настоящие время виртуальные миры широко используются в различных сферах, в том числе и в сфере образования. Только в виртуальном мире Second Life на сегодняшний момент насчитывается более 1000 представительств крупнейших университетов мира [1].

Одной из отличительных особенностей виртуальных миров является направленность на максимальное сходство восприятия человека виртуального трехмерного пространства с реальным миром. Это особенно полезно при совместном обучении

географически распределенных пользователей, т. к. позволяет обеспечить максимальную вовлеченность их в учебный процесс.

В виртуальных мирах актуальной проблемой является обеспечение осведомленности пользователей о действиях других пользователей, находящихся в одном и том же месте виртуального пространства. Эта задача решается путем синхронизации реплик (копий) синхронизируемых объектов виртуального мира на компьютерах пользователей. Эта задача усложняется тем, что синхронизация реплик трехмерных объектов требует передачи большего объема. Оптимизация сетевого трафика сообщений синхронизации объектов виртуального мира позволит снизить нагрузку на сетевой канал и обеспечить возможность его использование большим количеством пользователей в условиях России.

Образовательный виртуальный мир «Виртуальная академия» или кратко – vAcademia (http://www.vacademia.com) предназначена для проведения учебных занятий для географически распределённых пользователей в совместной трёхмерной среде. Он представляет собой тропический остров, на котором мире размещено более 50 учебных аудиторий. Пользователям предоставлены различные инструменты для организации совместной учебной работы: презентационные доски, аудио-конференции, система голосования, ретрансляция изображения с экрана монитора или веб-камеры и т. п.

В рамках этого проекта предложен подход к синхронизации трехмерных объектов виртуального мира, позволяющий существенно снизить трафик сообщений синхронизации. Настоящее исследование посвящено описанию этого подхода.

Объекты виртуального мира

Виртуальными мирами называют многопользовательские программные системы виртуальной реальности, основанные на трехмерной графике [2]. Отличительной особенность систем виртуальной реальности является то, что они погружают пользователя в пространство, созданное компьютером [3]. В системах виртуальной реальности, основанные на трехмерной графике, это достигается путем создания на экране компьютера привычного для человеческого восприятия трехмерного пространства и предоставления пользователю возможности перемещаться в этом пространстве и взаимодействовать с окружающими объектами [4], [5], [6].

Ричард Бартл, один из ведущих специалистов в области создания виртуальный миров, сформулировал определение виртуального мира в виде ряда принципов, которым он должен соответствовать [7]:

- принцип «внутренних законов»: у виртуального мира есть встроенные автоматические правила, по которым он действует;
- принцип «аватаров»: каждый пользователь представлен внутри мира индивидуальным персонажем;
- принцип «немедленной реакции»: действия в мире происходят в режиме реального времени;
- принцип «общей реальности»: виртуальный мир един для всех его пользователей, и каждый из них, находясь в одном месте, видит одно и тоже;
- принцип «постоянства»: мир существует независимо от наличия в нем пользователей.

Таким образом, в виртуальном мире множество географически распределенных пользователей взаимодействуют с одной виртуальной реальностью, связываясь через сеть Интернет. Для того чтобы обеспечить выполнение принципа «общей реальности», необходимо решить задачи свойственные системам поддержки совместной работы [8]. Одна из основных задач заключается в том, чтобы каждый пользователь в разделяемом пространстве мог видеть то же, что видят другие пользователи.

Решение этой проблемы сводится к синхронизации (обеспечению эквивалентности) реплик разделяемого пространства [9].

В виртуальном мире разделяемым пространством являются все трехмерные объекты виртуальной реальности, на которой он основан. Количество трехмерных объектов в пространстве виртуального мира исчисляет от десятков тысяч до многих миллионов. Сетевые сообщения для синхронизации реплик разделяемого пространства, состоящего из такого количества объектов, требуют передачи по сети больших объемов данных и составляют большую часть сетевого трафика. Например, в системные требования популярного виртуального мира Second Life входит наличие кабельного или DSL-соединения с Интернет [10]. Это означает, что скорость соединения должна быть не менее 256 кбит/с. Это минимальная скорость, при которой возможно подключится к Second Life. По мнению обычных пользователей [11], а также исследователей, изучающих трафик Second Life [12], для комфортной работы скорость подключения должна быть не менее 600-700 кбит/с.

Подходы к синхронизации в виртуальных мирах

Чтобы уменьшить сетевой трафик сообщений синхронизации объектов виртуального мира, необходимо определить, какие данные передаются по сети для обеспечения синхронизации реплик виртуального мира, поддерживаемых на компьютерах разных пользователей. Для этого рассмотрим некоторые подходы к синхронизации реплик виртуального пространства, используемые в трехмерных чатах, многопользовательских онлайновых играх и неигровых виртуальных мирах.

Простейшей реализацией виртуальных миров являются трехмерные чаты. Их основное назначение – коммуникации, поэтому они характеризуются статичной трехмерной средой, созданной разработчиками и неизменяемой пользователями. Большинство трехмерных чатов, как например Club Cooee [13], обеспечивают возможность перемещения в виртуальном пространстве и обмен текстовыми сообщениями.

Основной информацией, необходимой для синхронизации локальных реплик виртуального пространства, являются трехмерные координаты аватаров, представляющих пользователей, и жесты, выполняемые аватарами для передачи невербальных коммуникаций. Поэтому клиентская программа транслирует в сеть координаты аватара, которым управляет пользователь, а из сети получают координаты аватаров других пользователей. Трехмерные чаты требуют минимального сетевого трафика.

С развитием компьютерной графики виртуальные миры стали использоваться в онлайновых играх. Рассмотрим особенности онлайновых игр на примере разработанной компанией Blizzard Entertainment онлайновой игры World of Warcraft [15], которая по данным книги «Книги рекордов Гиннеса» является самой полярной в мире онлайновой игрой [14].

После запуска клиентская программа игры подключается к игровому серверу. Затем пользователь авторизуется. После чего с сервера запрашивается информация о состоянии виртуального мира и на компьютере пользователя выстраивается его локальная реплика, используемая для визуализации трехмерной среды, в которую пользователь погружается. После чего пользователь, управляя своим аватаром, путешествует по обширному миру, выполняет задания, сражается, общается с другими пользователями. При этом клиентская программа предает в сеть информацию о действиях производимых пользователем, а из сети получает данные об изменениях, которые вносятся в локальную реплику виртуального мира и отображаются на экране дисплея.

Исходя из целей игры и принципов Ричарда Бартла, в онлайновой игре World of Warcraft необходимо учитывать не только местоположение аватаров пользователей, но и их действия. Кроме того, виртуальный мир World of Warcraft населен мно-

жеством автономных персонажей, местоположение и действия которых также необходимо синхронизировать. Поэтому World of Warcraft требуют на много большего сетевого трафика для синхронизации локальных реплик виртуального пространства по сравнению трехмерными чатами. Согласно системным требованиям пользователи должны иметь постоянное соединение с Интернетом с минимальной скоростью 56 кбит/с. При этом рекомендуется использовать подключение со скоростью 128-256 кбит/с [16].

Однако в World of Warcraft не все объекты виртуального пространства синхронизируются через сеть. К таким объектам относятся следующие: земля, горы, реки, деревья, растения, постройки и другие объекты, формирующие трехмерное пространство виртуального мира, но не влияющие на ход игры. Их местоположение и поведение объектов предопределено их типом и структурой виртуального пространства, которые задаются либо в программном коде игры, либо данных, описывающих виртуальный мир. Например, нет необходимости синхронизировать местоположение деревьев и движение их ветвей под воздействием ветра, так это не может быть изменено пользователями и не влияет на ход игры.

Таким образом, в виртуальных мирах онлайновых игр среди множества всех объектов выделяются синхронизируемые и несинхронизируемые объекты. К синхронизируемым объектам относятся аватары, автономные персонажи и другие объекты, изменяемые в результате действий пользователей. К несинхронизируемым объектам относятся объекты, формирующие трехмерное пространство виртуального мира, но не влияющие на ход игры.

Разработчики проекта SecondLife [17] поставили перед собой цель разработать виртуальный мир, пользователи которого сами же его создают [18]. Зарегистрировавшись в SecondLife и запустив клиентскую программу, пользователь входит в огромный виртуальный мир. В этом мире пользователи осваиваются на своих территориях, как им нравится, – засаживают их лесами, украшают цветами, заполняют городами, острова окружают морями, не забывая населять водные пространства прекраснейшими рыбами. Таким образом, все окружение виртуального пространства создано пользователями SecondLife.

Таблица 1

Область применения	**Множество синхронизируемых объектов**	**Способ синхронизации**	**Скорость соединения с Интернетом**
Трехмерные чаты	Аватары	Местоположение аватаров и их жесты	56 Кбит/с
Онлайновые игры	аватары, автономные персонажи, изменяемые пользователями объекты	Местоположение и выполняемые действия	128-256 Кбит/с
Создаваемые пользователями виртуальные миры	Все объекты виртуального пространства	Местоположение, выполняемые действия, изменение формы	600-700 Кбит/с

Чтобы сделать это возможным, а также обеспечить выполнение упомянутых выше принципов Бартла, разработчикам пришлось использовать механизм синхронизации для всех объектов виртуального мира, а не только для небольшой его части, как это сделано в онлайновых играх. Ведь если все объекты виртуального мира созданы пользователями, то каждый из них в любой момент может быть изменен, и согласно принципам «общей реальности» и «немедленной реакции», эти изменения сразу же должны быть видны всем другим пользователям, находящимся в том же

месте. Кроме того, при таком подходе меняется и характер синхронизации: теперь необходимо синхронизировать не только местоположение и действия объектов виртуального мира, но и их форму. При этом сетевой трафик существенно увеличивается и поэтому, согласно системным требованиям SecondLife, минимальная скорость соединения с Интернетом должна составлять 256 кбит/с. Рекомендуемая скорость подключения для комфортной работы должна быть не менее 600-700 кбит/с.

Таким образом, существует три подхода к синхронизации в виртуальных мирах. Каждый из них определятся областью его применения. Они задают множество синхронизируемых объектов и способ их синхронизации. При этом каждый из них предъявляет свои требования к скорости соединения с Интернетом. Подходы к синхронизации представлены в таблице 1.

Наиболее приемлемым в условиях России является подход синхронизации объектов виртуального мира, применяемый в онлайновых играх. Это означает, что трехмерное окружение виртуального пространства должно быть статичным, а множество синхронизируемых объектов определяется аватарами и небольшим множеством автономных объектов, для каждого из которых синхронизируются их местоположение в виртуальном мире и выполняемые ими действия. Однако этот подход не дает пользователям возможности самим формировать виртуальное пространство. Кроме того, при таком подходе виртуальный мир должен быть статичным, что является неприемлемым для постоянно развивающегося мира, который создается в рамках проекта «Виртуальная академия». Поэтому в рамках этого проекта исследованы возможности сокращения необходимого сетевого трафика за счет использования собственного подхода к синхронизации объектов виртуального пространства.

Низкоуровневое состояние объекта

Чтобы каждый пользователь в виртуальном мире мог видеть то же, что видят другие пользователи, необходимо обеспечить, чтобы на компьютерах всех пользователей были одинаковые данные об объектах виртуального мира. Совокупность данных обо всех объектах виртуального мира будем называть репликой мира по аналогии с понятием «реплика разделяемого пространства», используемым в системах совместной деятельности [19].

Таким образом, синхронизация в виртуальном мире – это процесс обеспечения эквивалентности реплик виртуального мира на компьютерах всех пользователей. Реплика виртуального мира складывается из реплик трехмерных объектов, формирующих его. Следовательно, чтобы оптимизировать сетевой трафик сообщений синхронизации, необходимо определить характер данных, входящих в реплику трехмерного объекта.

Как уже отмечалось, виртуальными мирами называют многопользовательские программные системы виртуальной реальности, основанные на трехмерной графике. При этом на экране компьютера визуализируется трехмерное пространство. В настоящее время для визуализации трехмерных объектов чаще всего используются полигональные модели.

Полигональная модель – это совокупность вершин, ребер и полигонов [20]. Вершины соединены ребрами таким образом, что каждое ребро принадлежит не более чем двум многоугольникам. Ребра ограничиваются двумя вершинами, а полигоны замыкаются цепочкой из последовательно соединенных ребер. Каждое ребро служит границей двух смежных полигонов, а каждая вершина принадлежит, по крайней мере, двум ребрам. При визуализации полигоны разбивают на треугольники, соединяя их вершины дополнительными ребрами. Каждая вершина связывается с помощью текстурных координат с графическим изображением, которое используется для текстурирования объекта при визуализации.

Таким образом, в реплику трехмерного объекта входит описание полигональной модели и графический файл с текстурой. Описание полигональной модели обозначим через m, а текстуру – через T.

Изменение координат вершин приводит к изменению топологии полигональной модели и отображается при визуализации. Плавное изменение координат вершин в течение времени называется вертексной анимацией или морфингом.

В работе [21] исследуется возможность использования вертексной анимации для передачи по сети динамических трехмерных объектов в виде потока данных. Этот способ получил название «потоковое 3D» по аналогии с «потоковым видео». В работе предлагается на сервере сформировать поток данных, описывающих полигональные модели объектов трехмерной сцены, а также изменение координат их вершин. При визуализации этого потока клиентская программа создает динамичное изображение трехмерной сцены. Предлагается этот подход использовать для демонстрации интерактивных трехмерных презентаций, а также в многопользовательских системах.

Если же этот подход использовать для создания виртуального мира, то проблема синхронизации отпадает сама собой, так как при этом всем пользователям транслируется потоки данных, созданные с одной и той же трехмерной сцены, что позволяет автоматически получать эквивалентные реплики виртуального мира на компьютерах всех пользователей. Однако авторы работы о потоковом 3D показали, что для работы подобной системы необходим сетевой канал пропускной способностью 2 Гбит/с., что на текущий момент является неприемлемым для массового использования.

Существует другой подход анимирования трёхмерных объектов, названный скелетной анимацией [30]. При этом каждая вершина полигональной модели связывается как минимум с одной костью скелета. Скелет состоит из иерархически связанных между собой костей. Каждая кость может вращаться вокруг точки привязки к родительской кости. Согласно этому вращению перемещаются вершины, связанные с этой костью, и все дочерние кости, а следовательно и связанные с ними вершинами. Таким образом, изменяя с временем углы поворота костей, а затем пересчитывая координаты вершин полигональной модели, создается анимация трёхмерного объекта. При этом размер данных описания такой анимации во много раз меньше, т. к. необходимо хранить лишь углы поворота костей.

Объекты виртуального мира представляют собой трехмерные объекты, созданные на основе полигональных моделей с использованием подхода скелетной анимации. Совокупность характеристик, необходимых для визуализации трехмерного объекта, назовем его низкоуровневым состоянием и обозначим:

$$s = < p, r, m, b_m, b, T >, \quad (1)$$

где характеристики объекта обозначены следующим образом:

- p – мировые координаты – однозначно описывают местоположение объекта в пространстве;
- r – поворот – определяет его ориентацию;
- m – полигональная модель – описывает геометрическую форму трехмерного объекта;
- b_m – модель скелета – позволяет анимировать отдельные части объекта;
- b – положение костей скелета – задают взаимное расположение отдельных частей объекта;
- T – текстура – задает цвет, прозрачность и другие характеристики, используемые при визуализации трехмерного объекта.

Каждая клиентская программа хранит у себя реплику (копию) низкоуровневого состояния s_{rep} каждого синхронизируемого объекта и использует эти данные для

его визуализации. Таким образом, процесс синхронизации сводится к обеспечению эквивалентности реплик низкоуровневых состояний синхронизируемых объектов.

Синхронизация объектов виртуального мира с использованием низкоуровневого состояния объекта

Реплика низкоуровневого состояния синхронизируемого объекта в процессе своего существования проходит следующие этапы:

- Этап создания;
- Этап изменения;
- Этап удаления.

Этапы существования реплики низкоуровневого состояния синхронизируемого объекта назовём её жизненным циклом. Жизненный цикл реплики низкоуровневого состояния синхронизируемого объекта тесно связан с действиями пользователя и с этапами существования соответствующего объекта виртуального мира. Рассмотрим, каким образом каждый этап жизненного цикла реплики низкоуровневого состояния объекта отражаются на сетевом трафике сообщений синхронизации.

На этапе «Создание реплики низкоуровневого состояния» клиентская программа должна получить низкоуровневое состояние объекта полностью, так как до этого момента она не располагает никакой информацией об этом объекте. Подобная ситуация возникает в следующих случаях:

- в мир вошёл новый пользователь;
- в мире создался новый синхронизируемый объект.

Когда в мир входит новый пользователь, его клиентская программа должна получить реплики низкоуровневых состояний всех синхронизируемых объектов. При этом клиентские программы всех остальных пользователей должны получить реплики низкоуровневого состояния аватара вновь вошедшего пользователя. В случае же когда в мире создается новый объект, клиентские программы всех пользователей должны получить реплики низкоуровневого состояния этого объекта. После того, как клиентская программа получит реплику низкоуровневого состояния синхронизируемого объекта, она готова к его визуализации.

Чтобы оценить влияние этапа «Создание реплики низкоуровневого состояния» на сетевой трафик сообщений синхронизации, необходимо оценить нижнюю и верхнюю границы размера низкоуровневого состояния синхронизируемого объекта. Для оценки размера его составных частей можно использовать документацию популярных библиотек визуализации трёхмерной графики OpenGL [24] и Microsoft DirectX [23], стандарта MPEG-4 [22], а также работы исследователей, посвященные оптимизации описания 3D-объектов, [21], [25], [26], [27], [28], [29].

Координаты в трёхмерном мире векторами из трёх значений типа float, который представляется 4-мя байтами. Поворот также кодируется вектором из четырёх значений типа float. Размер этих векторов рассчитывается точно: координаты – 12 байт, поворот – 16 байт.

Размер данных, необходимых для представления полигональной модели, складывается из размера данных, описывающих вершины, и данных, задающих их взаимосвязи при формировании треугольников. Каждая вершина описывается следующими данными:

- координаты – вектор из трёх значений типа float – 12 байт,
- нормаль – вектор из трёх значений типа float – 12 байт,
- текстурные координаты – вектор из двух значений типа float – 8 байт,
  цвет – вектор из трёх значений типа byte – 3 байта.

Следовательно размер данных одной вершины равен 35 байт. Для кодирования треугольников полигональной модели используют алгоритмы сжатия, которые позволяют использовать 2-3 бита для кодирования одного треугольника. Таким образом,

размер данных полигональной модели полигональной модели вычисляется по формуле:

$$size(m) = 35 \cdot n + \frac{3}{8} \cdot l, \quad (2)$$

где n – это количество вершин, а l – это количество треугольников.

Полигональная модель аватара средней детализации состоит из 3000 вершин и 2500 треугольников. Размер его полигональной модели можно оценить следующим образом: $size(m) = 35 \cdot 3000 + \frac{3}{8} \cdot 2500 = 105938$ байт.

В модели скелета трёхмерного объекта каждая вершина полигональной модели привязывается как минимум к одной кости. Поэтому размер модели скелета трёхмерного объекта прямо пропорционально зависит от количества вершин полигональной модели и сложности скелета. Каждая из костей скелета описывается следующими характеристиками:

- номер родительской кости – 1 байт,
- длина кости – значение типа float – 4 байта,
- поворот относительно родительской кости вектором из четырёх значений типа float – 16 байт.

Таким образом, нижнюю границу размера модели скелета трёхмерного объекта можно вычислить по формуле:

$$size(b_m) = 21 \cdot k + n \cdot 5, \quad (3)$$

где k – это количество костей скелета, n – это количество вершин полигональной модели.

Для обеспечения естественной подвижности аватаров при моделировании их движений используют скелеты из 50 костей. Размер модели скелета оценивается следующим образом: $size(b_m) = 21 \cdot 50 + 3000 \cdot 5 = 17050$ байт.

При появлении трёхмерного объекта в мире необходимо задать начальное положение костей его скелета. Оно задаётся путём определения поворота для каждой кости относительно родительской кости, который представляется вектором из четырёх значений типа float, что составляет 16 байт. Следовательно, нижняя граница размера данных положения костей скелета вычисляется по формуле:

$$size(b) = 16 \cdot k, \quad (4)$$

где k – это количество костей скелета.

Для аватара с моделью скелета из 50 костей $size(b) = 16 \cdot 50 = 800$ байт.

Для представления материала и текстуры используется растровое изображение со средним качеством. Его размер зависит от сложности и детальности трёхмерного объекта, что невозможно рассчитать формально. Поэтому в таблице 2 представлена нижняя оценка размера материала и текстуры для аватара с минимальным уровнем детализации – ~150 Кбайт.

В таблице 2 приведены оценки размеров всех компонентов размер низкоуровневого состояния синхронизируемого объекта на этапе «Создание реплики низкоуровневого состояния». Общий размер низкоуровневого состояния при этом равен примерно 274 Кбайт. При скорости интернет соединения 512 Кбит/с это требует 4,3 секунды для передачи этих данных с сервера клиентской программе для каждого аватара.

На этапе «Изменение реплики низкоуровневого состояния» синхронизируемый трёхмерный объект существует в виртуальном мире. При этом он может совершать следующие действия:

- перемещение в пространстве – изменяются его координаты и поворот;
- совершение движений – изменяются положения костей скелета;

- изменение формы – изменяется топология полигональной модели и структура скелета;
- изменение текстуры – заменяется одно растровое изображение на другое.

Таблица 2

Часть низкоуровневого состояния	**Тип данных**	**Оценка размера данных**
p – мировые координаты	3 числа типа float	12 байт
r – поворот	4 числа типа float	16 байт
m – полигональная модель	Для аватара средней детализации используется полигональная модель из 3000 вершин. Нижняя граница размера данных вычисляется по формуле (2)	~106 Кбайт
b_m – модель скелета	Для аватара используется скелет из 50 костей. Нижняя граница размера данных вычисляется по формуле (3).	~17 Кбайт
b – положение костей скелета	Для аватара используется скелет из 50 костей. Нижняя граница размера данных вычисляется по формуле (4).	800 байт
T – материал и текстуры	Используется растровое изображение со средним качеством	~150 Кбайт
	Итого	~274 Кбайт

Каждое действие или изменение синхронизируемого объекта влечет изменение его низкоуровневого состояние. Чтобы обеспечить эквивалентность реплик низкоуровневого состояния на компьютерах всех пользователей необходимо при каждом изменении передавать всё низкоуровневое состояние или лишь его изменённую часть. Передавать всё низкоуровневое состояние на каждое его изменение невозможно, поэтому необходимо определить из чего складывается Δs – данные, которые необходимо добавить к низкоуровневому состоянию синхронизируемого трёхмерного объекта, чтобы получить его следующее состояние. Назовём Δs изменением низкоуровневого состояния синхронизируемого объекта, которая задаётся следующим образом:

$$\Delta s = s_{n+1} - s_n , \tag{5}$$

где s_n, s_{n+1} - соответственно текущее и следующее состояние синхронизируемого трёхмерного объекта. Таким образом, изменения низкоуровневого состояния складываются из изменения его составляющих по формуле (1).

Для примера рассмотрим аватара. Он может перемещаться по виртуальному миру, поэтому изменение величин Δp и Δr войдут в изменение состояния синхронизируемого объекта.

При совершении аватаром движений изменяется положение костей его скелета. Для обеспечения естественного поведения аватара в виртуальном мире необходимо, чтобы он постоянно двигался, даже если он никуда не перемещается. При этом изменяется положение его костей, однако модель скелета остаётся неизменной. Поэтому изменение положения костей Δb является составляющей изменения состояния синхронизируемого объекта.

Изменения формы и текстуры происходят очень редко. Это означает, что изменяются полигональная модель, модель скелета и текстура, что равносильно созданию трёхмерного объекта заново. Поэтому в этом исследовании эти действия не рассматриваются.

Исходя из выше сказанного, изменение низкоуровневого состояния можно представить как совокупность изменений его характеристик:

$$\Delta s = < \Delta p, \Delta r, \Delta b > \quad (6)$$

Чтобы определить влияние этапа «Изменение реплики низкоуровневого состояния» на сетевой трафик сообщений синхронизации, необходимо оценить размер данных изменения состояния накопленных в течение одной секунды. Размер этих данных вычисляется по формуле:

$$size(\Delta s) = size(\Delta p) + size(\Delta r) + size(\Delta b) \quad (7)$$

Чтобы перемещение аватара было плавным клиентская программа как минимум 10 раз в секунду должна получать координаты и поворот аватара. То есть $size(\Delta p) = 12 \cdot 10 = 120$ байт/с, и $size(\Delta r) = 16 \cdot 10 = 160$ байт/с. Чтобы обеспечить естественную подвижность аватара необходимо постоянно изменять положение костей его скелета. Это означает, что в течение секунды каждая кость скелета аватара изменит своё положение как минимум один раз. То есть $size(\Delta b) = 800$ байт/с. Таким образом, для передачи изменения низкоуровневого состояния каждого аватара необходимо $size(\Delta s) = 120 + 160 + 800 = 1080$ байт/с или 8,4 Кбит/с.

Этап «Удаление реплики низкоуровневого состояния» требует всего лишь одного сообщения, поэтому его влиянием на сетевой трафик сообщений системы синхронизаций можно пренебречь.

Таким образом, при использовании подхода синхронизации через низкоуровневые состояния для каждого аватара необходимо на этапе создания реплики состояния передать с сервера 273 Кбайт данных, а для обеспечения существования аватара в виртуальном мире необходим трафик не менее 8 Кбит/с. Для обеспечения присутствия в одном месте виртуального мира 20 (количество учащихся на одном занятии) пользователей для системы синхронизации необходим канал шириной не менее 160 Кбит/с. На учебных занятиях виртуального мира vAcademia может присутствовать в несколько раз больше пользователей.

Прикладное состояние синхронизируемого объекта

Существуют различные подходы уменьшения сетевого трафика. Как правило, они связаны с использованием различных алгоритмов сжатия. Однако выигрыш при этом получается небольшой. Для получения большего эффекта необходимо изменить подход к синхронизации реплик состояний трёхмерных объектов виртуального мира.

Использование низкоуровневого состояния для синхронизации объектов виртуального мира позволяет серверу осуществлять полный контроль над существованием этих объектов. На сервере не только хранятся эталонные реплики состояний объектов, но и выполняются сценарии, описывающие их поведение, а также рассчитывается ограничения физической модели виртуального мира, как например это сделано в SecondLife. При этом достигается точное воспроизведение всех взаимодействий синхронизируемых объектов в виртуальном мире. Однако за это приходится платить повышением вычислительной нагрузки на сим (сервер одного региона). Причём если сценарий поведения какого-либо объекта требует больших вычислений, перегружающих сим, то это ощущают все подключенные к нему пользователи. В то же время мощность современных компьютеров конечных пользователей достаточна не только для того, чтобы отображать трёхмерное пространство на экране монитора, но и вполне хватает для того, чтобы выполнять сценарии поведения и рассчитывать ограничения физической модели для объектов в зоне видимости.

Таким образом, хранение на сервере низкоуровневого состояния всех объектов становится необязательным. Это позволяет использовать для обеспечения эквивалентности реплик состояния синхронизируемых объектов данные, отличные от низкоуровневого состояния. Эти данные должны нести минимум информации, необхо-

димой для перевода объекта из одного состояния в другое. Так как для того чтобы определить характер этих данных необходимо проанализировать область применения каждого синхронизируемого объекта, поэтому они названы прикладным состоянием.

Прикладное состояние синхронизируемого трёхмерного объекта определяется следующим образом: данные, представляющие состояние синхронизируемого трёхмерного объекта как функционального объекта виртуального мира, называются прикладным состоянием.

Чтобы определить характер данных, включенных в прикладное состояние каждого объекта виртуального мира, необходимо проанализировать его функциональность и выявить ключевые действия, которые он выполняет. Для примера проанализируем функциональность аватара. Во время своего существования в виртуальном мире аватар выполняет следующие действия:

- стоит, при этом аватар находится на одном месте в стоячей позе, его координаты и поворот не изменяются;
- идёт, при этом аватар перемещается в трехмерном пространстве, изменяются его координаты, поворот и воспроизводится анимация ходьбы;
- бежит, при этом аватар перемещается в трехмерном пространстве, изменяются его координаты, поворот и воспроизводится анимация бега;
- сидит, при этом аватар находится на одном месте в сидячей позе, его координаты и поворот не изменяются;
- отображает реплику из чата, при этом над аватаром появляется облако с текстовым сообщением, написанным пользователем в чате;
- выполняет жест, при этом аватар выполняет одно из набора предопределенных движений по выбору пользователя, воспроизводится анимация, отображающая этот жест;
- бездействует, при этом аватар выполняет случайную последовательность движений для придания естественности его поведению, воспроизводится анимация, отображающая их.

Чтобы от действий, выполняемых аватаром, перейти к прикладным состояниям необходимо представить его функциональность в виде конечного автомата. Это позволяет определить возможные последовательности переходов аватара из одного состояния в другое, то есть смоделировать все изменения состояний объекта как его реакцию на действия пользователя или другие события виртуального мира.

Исходя из функциональности аватара, выделяются следующие состояния конечного автомата, в виде которого его можно представить:

- «Создание аватара» – соответствует этапу создания реплики низкоуровневого состояния аватара;
- «Существование аватара» – соответствует этапу изменения реплики низкоуровневого состояния аватара;
- «Удаление аватара» – соответствует этапу удаления реплики низкоуровневого состояния аватара.

При этом состояние «Существование аватара» детализируется следующими подсостояниями:

- «Стоит» – аватар стоит;
- «Идёт» – аватар идёт;
- «Бежит» – аватар бежит;
- «Сидит» – аватар сидит;
- «Молчит» – облако над головой аватара скрыто;
- «Говорит в чат» – над головой аватара отображается облако с текстом;
- «Бездействует» – аватар выполняет случайную последовательность движений;
- «Выполняет жест» – аватар выполняет предопределенное движение.

Чтобы добиться естественности поведения аватара необходимо, чтобы некоторые действия выполнялись параллельно с другими. Например, жест поднятия руки для приветствия он должен выполнять и стоя, и сидя, во время перемещения. Таким образом, состояния аватара делятся на два типа:

- совместные – состояния, в которых аватар может находиться одновременно, например, состояния «Идет» и «Говорит в чат»;
- несовместные – состояния, в которых аватар не может находиться одновременно, например, он не может быть одновременно в состояниях «Сидит» и «Идет».

Таблица 3

	Стоит	**Сидит**	**Идет**	**Бежит**	**Говорит в чат**	**Молчит**	**Выполняет жест**	**Бездействует**
Стоит		Нет	Нет	Нет	Да	Да	Да	Да
Сидит	Нет		Нет	Нет	Да	Да	Да	Да
Идет	Нет	Нет		Нет	Да	Да	Да	Да
Бежит	Нет	Нет	Нет		Да	Да	Да	Да
Говорит в чат	Да	Да	Да	Да		Нет	Да	Да
Молчит	Да	Да	Да	Да	Нет		Да	Да
Выполняет жест	Да	Да	Да	Да	Да	Да		Нет
Бездействует	Да	Да	Да	Да	Да	Да	Нет	

В таблице 3 представлена матрица совместности состояний аватара. В каждой ячейке на пересечении строк и столбцов поставлены отметки «Да» или «Нет». Отметка «Да» означает, что состояния совместны, а отметка «Нет» означает, что состояния несовместны.

Для моделирования конечных автоматов используется UML-диаграмма состояний, в которой задаются возможные последовательности переходов из одного состояния в другое. Для моделирования конечных автоматов, имеющих совместные и несовместные состояния, применяются параллельные вложенные и последовательные вложенные состояния [31]. На рисунке 1 представлена UML-диаграмма состояний аватара.

На этой диаграмме состояние «Существование аватара» представлено суперсостоянием, в состав которого входят его подсостояния. Причем совместные состояния представлены в виде параллельных вложенных состояний, и отделены друг от друга пунктирной линией. Параллельные вложенные состояния, по сути, являются независимыми конечными подавтоматами. Поэтому смена состояния одного из этих подавтоматов никак не отражается на состоянии других.

Таким образом, в прикладное состояние аватара описывается как совокупность трёх объектов, каждый из которых описывает состояние одного подавтомата, отвечающего за отдельный аспект функциональности аватара:

$$S = < P, M, C > \quad (8)$$

где P – объект, описывающий состояние подавтомата перемещения аватара в виртуальном пространстве, M – объект, описывающий выполнение аватаром движений, C – объект, описывающий облако над головой аватара. Эти объекты имеют подобную структуру. Они состоят из двух частей:

- идентификатор состояния, в которое аватар должен перейти;

- параметры, необходимые клиентской программе для перевода аватара в заданное состояние.

Исходя из этого, при изменении состояния аватара клиентская программа обменивается с сервером информацией об изменении состояния лишь одного подавтомата. Эта информация названа пакетом обновления прикладного состояния.

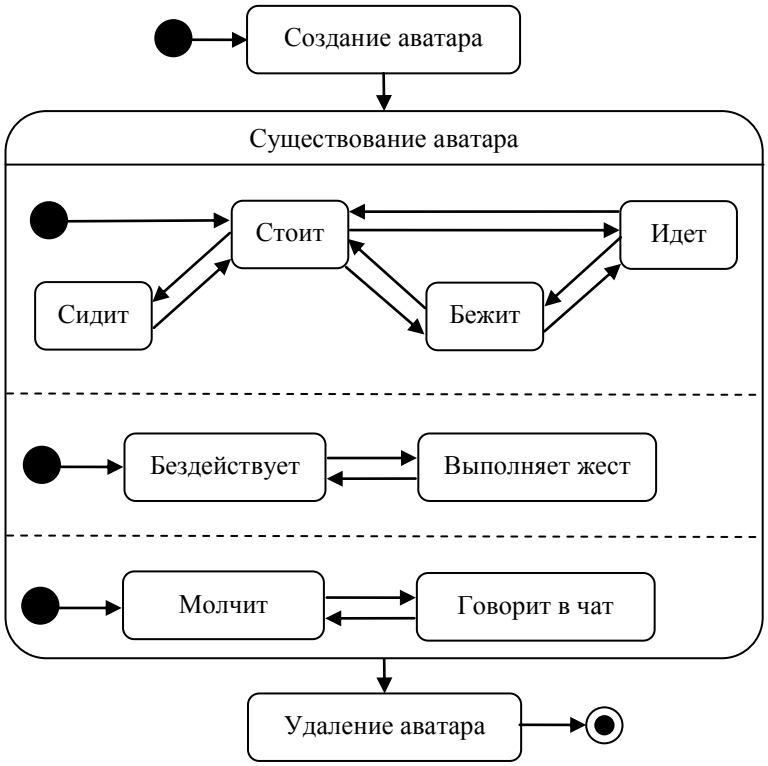

Рисунок 1. UML-диаграмма состояний аватара

Таким образом, клиентская программа производит все необходимые вычисления для реализации поведения аватара. При каждом действии пользователя клиентская программа определяет изменение прикладного состояния этих аватара, формирует пакет обновления и затем отправляет его на сервер. Сервер рассылает этот пакет обновления всем подключенным к нему компьютерам пользователей. Клиентская программа другого пользователя, получив от сервера пакет обновления чужого аватара, изменяет его прикладное состояние и запускает процедуру воссоздания его низкоуровневого состояния. Как только низкоуровневое состояние обновляется, пользователь видит на экране действия аватара.

Синхронизация объектов виртуального мира с использованием прикладного состояния объекта

Применение прикладного состояния объекта не влияет на количество данных, передаваемых с сервера на этапе создания аватара. В тоже время этот подход позволяет сократить объем передаваемых данных, необходимых для синхронизации реплик его состояния на этапе существования аватара в виртуальном мире. Это обусловлено тем, что изменение прикладного состояния, т. е. смена действий аватара, происходит намного реже, чем изменения положения костей скелета в скелетной анимации.

Исходя из выше сказанного и состава прикладного состояния (8), оценка трафика сетевых сообщений синхронизации с использованием подхода синхронизации реплик состояния аватара через его прикладные состояния будет рассчитываться в виде:

$$size(\Delta S) = \overline{size(P)} \cdot p(P) + \overline{size(M)} \cdot p(M) + \overline{size(C)} \cdot p(C) \quad (9)$$

где $\overline{size(P)}$ – средний размер параметров, описывающих состояние подавтомата перемещения аватара, $p(P)$ – вероятность перемещения аватара в течение одной секунды, $\overline{size(M)}$ – средний размер параметров, описывающих состояние подавтомата выполнения движений, $p(M)$ – вероятность выполнения заданного пользователем движения в течение одной секунды, $\overline{size(\Delta C)}$ – средний размер параметров, описывающих смену состояния облака над головой аватара, $p(C)$ – вероятность смены содержимого в течение одной секунды.

Таким образом, необходимо определить средний размер параметров, необходимых для смены состояний и вероятность возникновение этих состояний в течение одной секунды для каждого подавтомата. В таблице 4 представлены эти расчёты.

Таблица4

Подавтомат	**Состояние**	**Размер параметров**	**Средний размер параметров**	**Количество за 1 час**	**Вероятность в течение 1 секунды**
Перемещение аватара, *P*	Стоит	29 байтов	32,5 байта	180	0,0500
	Идёт	25 байтов			
	Бежит	25 байтов			
	Сидит	51 байт			
Выполнение движений, *M*	Бездействует	1 байт	26 байтов	60	0,0167
	Выполняет жест	51 байт			
Облако над головой, *C*	Молчит	1 байт	16 байтов	60	0,0167
	Говорит в чат	31 байт			

Средний размер параметров при перемещении аватара составляет 32,5 байта. При средней активности пользователя за 1 час его аватара совершает около 180 перемещений. При этом пользователь общается с другими пользователями. Он показывает своё отношение к происходящему жестами, выполняя до 60 движений за час, на каждое из которых необходимо передать в среднем 26 байтов данных. Средний размер параметров для отображения сообщений чата в облаке над головой аватара составляет 16 байтов, при этом один пользователь за час отправляет до 60 сообщений.

Таким образом, расчеты по формуле (9) показывают, что оценка трафика сетевых сообщений синхронизации с использованием подхода синхронизации реплик состояния аватара через его прикладные состояния равна $size(\Delta S) = 32{,}5 \cdot 0{,}05 + 26 \cdot 0{,}0167 + 16 \cdot 0{,}0167 = 2{,}325$ байта. Это составляет 18,6 бит/с, что почти в 450 раз меньше чем в случае использования низкоуровневых состояний для синхронизации реплик состояния аватара. Для обеспечения присутствия в одном месте виртуального мира 25 пользователей для системы синхронизации необходим канал шириной менее 0,5 Кбит/с. Это позволяет обеспечить присутствие на учебных занятиях виртуального мира vAcademia в несколько раз больше пользователей.

В виртуальном мире vAcademia описанный подход синхронизации объектов через их прикладные состояния используется не только для аватаров, но и для всех синхронизируемых объектов. К таким объектам относятся презентационные доски, позволяющие отображать на них слайды презентации, рисовать, размещать растровые изображения, ретранслировать изображение с экрана монитора или веб-камеры.

Кроме того, к этой же категории относятся создаваемые пользователем объекты, которые он выбирает из большой библиотеки и размещает в виртуальном мире. Библиотека объектов включает разнообразные объекты интерьера: переносные презентационные доски, стулья кресла, диваны, скамейки и т.п. Каждый объект, внесенный в виртуальный мир пользователем, становится доступен всем пользователя, и может быть использован ими согласно его функциональности.

Благодаря применению подхода синхронизации объектов через их прикладные состояния весь виртуальный мир vAcademia обслуживается одним сервером, несмотря на его размер. В виртуальном мире vAcademia насчитывается более 50 учебных аудиторий, расположенных на одном острове размером примерно 800x800 метров. В виртуальном мир SecondLife для поддержки такого острова необходимо использовать 9 регионов, каждый из которых представляет собой один сервер.

Кроме того, каждый пользователь, путешествуя по острову, может видеть, как проходят занятия сразу в нескольких аудиториях, так как учебные аудитории либо размещены в зданиях с полупрозрачными стенами, либо на открытых площадках. Это обеспечивает эффект присутствия даже для пользователей, которые не находятся на занятиях.

Заключение

Как показывает настоящие исследование, применение подхода синхронизации трёхмерных объектов на основе их прикладных состояний позволяет снизить сетевого трафик сообщений синхронизации объектов более чем в 450 раз. Это позволяет достичь следующих практических результатов:

- уменьшить требования к пропускной способности Интернет-соединения конечного пользователя;
- улучшить качество сервисов, реализующих инструменты аудио-конференций и ретрансляции изображения с экрана монитора или веб-камеры, за счёт передачи им части пропускной способности канала, ранее использованной сообщений синхронизации;
- уменьшить нагрузку на сервер, обеспечивающий синхронизацию реплик состояний трёхмерных объектов, и тем самым увеличить размер виртуального мира, поддерживаемого одним сервером.

Опыт использования предложенного подхода в виртуальном мире vAcademia показал, что он нуждается в доработке для объектов, создаваемых пользователями, поведение которых определяется ими же с помощью специального языка. Наши следующие исследования будут проводиться в этом направлении.

Литература

1. Gregory S., Tynan B., Introducing Jass Easterman: My Second Life learning space. // Proceedings ascilite Auckland 2009, pp. 377-386 URL: http://www.ascilite.org.au/conferences/auckland09/procs/gregory.pdf

2. Vellon M, Marple K, Mitchell D, Drucker S. The architecture of a distributed virtual worlds system. Proceedings of the 4th USENIX Conference on Object-Oriented Technologies and Systems (COOTS), Santa Fe, New Mexico, April 27-30. Usenix Association: Berkeley, CA, 1998.

3. NCAC (National Center on Accessing the General Curriculum) 2003. Virtual Reality/Computer Simulations: Curriculum Enhancement. U.S. Office of Special Education Programs.
4. Brooks F. P. Walkthrough – A Dynamic Graphics System for Simulating Virtual Buildings – Proceedings of 1986 Workshop on Interactive 3D Computer Graphics (Chapel Hill, North Carolina). Computer Graphics. pp. 9-21. 1986.
5. Brooks F. P. Grasping Reality Through Illusion: Interactive Graphics Serving Science. Proceedings of SIGCHI - 1988. pp 1-11. 1988.
6. Sutherland I.E. The Ultimate Display. Information Processing. Proc. IFIP Congress 65. 506-508. 1965.
7. Bartle R. A. Designing virtual worlds. // New Riders, 2004. ISBN: 0131018167, 9780131018167.
8. Antunes P., Guimaraes N. Multiuser Interface Design in CSCW Systems, BROADCAST Technical Report Series No. 71, ESPRIT Basic Research Project 6360, 1994.
9. Antunes P., Guimaraes N. A Distributed Model and Architecture for Interactive Cooperation // Proceedings of the Fourth Workshop on Future Trends of Distributed Computing Systems, IEEE Computer Society Press, Lisbon, Portugal, 1993, pp. 143-149, ISBN: 0-8186-4430-3.
10. Second Life. System Requirements. URL: http://secondlife.com/support/sysreqs.php (дата обращения: 20.06.2011)
11. MMOGame навигатор в мире онлайн игр. URL: http://mmogame.ru/index/game/SecondLife/ (дата обращения: 12.05.2011)
12. Fernandes S., Kamienski C., Sadok D., Moreira J., Antonello R. Traffic Analysis Beyond This World: the Case of Second Life // NOSSDAV, Illinois, USA, June 2007.
13. Интерактивный 3D чат & Виртуальный 3D мир. - Club Cooee. URL: http://ru.clubcooee.com/ (дата обращения: 26.06.2011)
14. Guinness World Records Gamer's Edition - Records - PC Gaming. URL: http://gamers.guinnessworldrecords.com/records/pc_gaming.aspx (дата обращения: 22.07.2010)
15. World of Warcraft Europe. URL: http://www.wow-europe.com/ru/index.xml (дата обращения: 02.04.2011)
16. World of Warcraft Europe. Системные требования. URL: http://www.wow-europe.com/ru/requirements/technical.html (дата обращения: 02.04.2011)
17. Second Life Official Site. URL: http://secondlife.com/ (дата обращения: 22.05.2011)
18. Леско М. Империя чувств // "Профиль": журнал. – Москва: 2008. – N 23. –с. 66-70
19. Greif I. Computer Supported Cooperative Work: A book of readings. – Morgan Kaufman Publishers, California, 1988.
20. Shirley P., Marschner S. и др., Fundamentals of Computer Graphics, Third Edition – A. K. Peters, 2009. ISBN: 9781568814698
21. Olbrich S., Pralle H., Virtual reality movies-real-time streaming of 3D objects, Computer Networks: The International Journal of Computer and Telecommunications Networking, v.31 n.21, p.2215-2225, Nov. 1999
22. Signès, J., Fisher, Y., Eleftheriadis, A., MPEG-4's binary format for scene description, SP:IC(15), No. 4-5, January 2000, pp. 321-345.
23. Microsoft DirectX SDK Documentation. URL: http://msdn.microsoft.com/en-us/library/ee663274(VS.85).aspx (дата обращения: 12.06.2011)
24. OpenGL(R) Reference Manual : The Official Reference Document to OpenGL,Version 1.4 (4th Edition) by OpenGL Architecture Review Board. ISBN 032117383
25. Peng J., Eckstein I., Jay Kuo C.-C. A novel and efficient progressive lossless mesh coder, In ACM SIGGRAPH Technical Sketches, 2006.
26. Bajaj C.L., Pascucci V., Zhuang G., Single Resolution Compression of Arbitrary Triangular Meshes with Properties // Computational Geometry: Theory and Applications, Vol. 14, No. 1--3, pp. 167--186. 1999.

27. Taubin G. 3D geometry compression and progressive transmission // EUROGRAPH-ICS 99. — 1999.
28. Isenburg, M., Snoeyink, J. Face fixer: compressing polygon meshes with properties. // SIGGRAPH, 2000, 263-270
29. Rossignac J., Edgebreaker: Connectivity compression for triangle meshes // IEEE Trans. Visualization and Computer Graphics, vol. 5, no. 1, pp. 47-61, 1999.
30. Kavan L., Zara J. Real time skin deformation with bones blending. // WSCG Short Papers Proceedings. Citeseer, 2003. URL: http://wscg.zcu.cz/wscg2003/Papers_2003/G61.pdf
31. Hennicker R., Koch N. Systematic Design of Web Applications with UML. // Siau, K. & Halpin, T., (Eds.) Unified Modeling Language: Systems Analysis, Design and Development Issues, Hershey, PA: Idea Group Publishing, 1-20.