ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ НАУКИ
ИНСТИТУТ АВТОМАТИКИ И ПРОЦЕССОВ УПРАВЛЕНИЯ
ДАЛЬНЕВОСТОЧНОГО ОТДЕЛЕНИЯ
РОССИЙСКОЙ АКАДЕМИИ НАУК

На правах рукописи

Федорищев Леонид Александрович

МОДЕЛИ, МЕТОДЫ И ИНСТРУМЕНТАЛЬНЫЕ СЕРВИСЫ ДЛЯ СОЗДАНИЯ ПРОФЕССИОНАЛЬНЫХ ВИРТУАЛЬНЫХ ОБЛАЧНЫХ СРЕД

05.13.11 – математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель –
д.т.н. В.В. Грибова

Владивосток
2013

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы. В настоящее время виртуальные среды активно используются в образовании, военной области, медицине, индустрии развлечений, бизнесе, инженерии, спорте, социальных сферах. Виртуальная среда – это компьютерная модель фрагмента реального мира, имитирующая процессы явления или события, которые могут в нем происходить под воздействием различных факторов. По типу взаимодействия с пользователем виртуальные среды могут быть разделены на неинтерактивные и интерактивные. Неинтерактивные среды предназначены для наблюдения, демонстрации каких-либо явлений или процессов, пользователь в таких средах выступает в качестве зрителя; в интерактивных средах пользователь может активно взаимодействовать с элементами среды, изменяя ее. По требованиям к профессиональной подготовке их пользователей виртуальные среды можно разделить на среды общего назначения и профессиональные. Характерными особенностями виртуальных сред общего назначения (типичный представитель - компьютерные игры) является отсутствие каких-либо требований к профессиональной подготовке их пользователей; в них заранее определяется логика работы приложения, которая, как правило, в процессе использования мало подвержена изменениям; для таких сред характерна также коммерческая направленность реализации, что позволяет привлечь достаточно большой бюджет к разработке таких сред. Профессиональные виртуальные среды предназначены для решения задач в некоторой предметной области подготовленными специалистами, их основу составляют предметные знания, которые подвержены частым изменениям, а бюджет при их разработке, за редким исключением, очень мал.

В общем случае процесс создания виртуальной среды состоит из нескольких основных этапов: разрабатываются 3D-модели объектов виртуального мира, для большинства моделей объектов описываются сценарии их поведения и возможного изменения отображения в виртуальной среде; затем из разработанных объектов формируется виртуальная среда (виртуальное

окружение) – определяется положение объектов относительно друг друга, их размер, повороты и другие необходимые атрибуты; отдельной трудоемкой задачей является описание возможных сценариев как влияния объектов друг на друга, так и изменения виртуального мира при воздействии на него пользователя; функционал некоторых интерактивных сред требует включения оценки действий пользователя при его взаимодействии с виртуальным миром.

На сегодняшний день существуют различные специализированные и универсальные инструментальные средства, пакеты прикладных программ, библиотеки для создания виртуальных сред общего назначения: Дельфин, ToolBook, Lectora, CAVE, WorldToolKit, Avango, Lightning, Juggler, Unity3D, Virtools, Alternativa3D, Flare3D и многие другие. Значительный вклад в разработку и исследование методов и средств создания виртуальных сред внесли российские и зарубежные ученые: Беневоленский С.Б., Бобков В.А., Борзых А.А., Борисов В.Г., Вавилова Н.И., Валькман Ю.Р., Гаммер М.Д., Данилова С.К., Дзюбенко О.Л., Донской А.Н., Коженков А.О., Марченко А.Л., Сук А.Ф., Трухин А.В., Филатова Н.И., Чинакал В.О., Bierbaum A., Craig A., Cruz-Neira C., Erlbaum L., Ghee S., Jacko J., Just C., Hale K., Hansen K., Sears A., Stanney K., Sherman W., Vincenti G., Will J. и другие.

Особенностью профессиональных виртуальных сред является то, что они содержат предметные знания; их носителями являются эксперты предметной области, которые должны эти знания формировать и сопровождать. В архитектуре таких систем выделяется специализированный компонент (база знаний) и используется специализированная технология разработки, включающая экспертов и поддерживаемая специализированным инструментарием.

Однако в настоящее время неизвестны специализированные средства для разработки именно профессиональных виртуальных сред. Использование инструментов общего назначения для создания профессиональных виртуальных сред делает процесс их разработки и особенно сопровождения чрезмерно трудоемким и дорогим. Все имеющиеся средства ориентированы на

использование программистами, иногда совместно с дизайнерами; процесс разработки связан с программированием нетривиальных скриптов или программ на языках программирования с последующей сборкой и компиляцией. Часто для разработки требуется использовать несколько различных библиотек и инструментальных средств и затем собирать из них единую систему. Включение в процесс разработки экспертов предметной области возможно только в качестве консультантов, а не полноправных его участников. Любое изменение требует трудоемкого перепрограммирования, последующей сборки и компиляции виртуальной среды. Отдельной проблемой является обеспечение широкой доступности через Интернет как средств разработки и сопровождения таких виртуальных сред, так и готовых реализаций.

Указанные выше факторы определяют актуальность теоретических и прикладных исследований диссертации, направленных на решение проблемы создания и сопровождения профессиональных виртуальных облачных сред.

Целью диссертационной работы является разработка моделей, методов и инструментальных сервисов для создания и сопровождения профессиональных виртуальных облачных сред.

Для достижения поставленной цели необходимо решить следующие **задачи**:

1. Разработать общие принципы разработки инструментального сервиса, обеспечивающего создание, функционирование и сопровождение профессиональных виртуальных облачных сред.
2. Разработать онтологию профессиональной виртуальной среды и основанную на ней декларативную модель профессиональной виртуальной среды.
3. Разработать методы интерпретации декларативной модели профессиональной виртуальной среды.
4. Разработать методы реализации инструментальных сервисов для проектирования, интерпретации и сопровождения профессиональных виртуальных облачных сред.

5. Разработать технологию проектирования и сопровождения профессиональных виртуальных облачных сред и выполнить ее экспериментальную проверку.

Методы исследования. В работе использовались методы, базирующиеся на аппарате теории множеств, компьютерной графики, системного анализа, искусственного интеллекта, объектно-ориентированного анализа и проектирования, а также методы веб-программирования.

Научная новизна работы состоит в следующем:

1. Предложены принципы создания и сопровождения профессиональных виртуальных облачных сред, в соответствии с которыми проектирование, реализация и сопровождение такой среды заменяется проектированием и сопровождением ее декларативной модели с последующей интерпретацией; этап кодирования на языке программирования в зависимости от конкретной виртуальной среды либо сводится к минимуму, либо отсутствует.
2. Разработана онтология профессиональной виртуальной среды, состоящая из трех основных компонентов: объектов виртуального мира; действий, которые можно производить с объектами виртуальной среды; сценария возможных действий пользователя для получения результата, определяемого обучающим заданием. В онтологии явно выделено два уровня: логический и презентационный.
3. Предложены различные типы моделей профессиональных виртуальных облачных сред, которые являются, во-первых, декларативными, во-вторых, формируются по онтологии, в-третьих, благодаря наличию логического и презентационного уровня в онтологии позволяют не только включить в разработку и сопровождение декларативной модели экспертов предметной области и дизайнеров, но также и разделить работу между ними.
4. Впервые разработан метод интерпретации декларативной модели виртуальной среды, заключающийся в определении способа построения

виртуальной сцены, определении схем работы клиентской и серверной частей, обработки событий и сообщений, способов хранении данных.

Практическая ценность и реализация результатов работы.

Практическая значимость полученных в диссертационной работе результатов заключается в том, что разработанный комплекс программных средств (облачных сервисов) позволяет создавать и сопровождать профессиональные виртуальные облачные среды на основе декларативного подхода с включением в процесс разработки экспертов предметной области.

Создана технология проектирования и сопровождения профессиональных виртуальных сред с использованием разработанных программных средств.

На основе предложенной технологии с помощью разработанного комплекса программных средств реализованы профессиональные виртуальные облачные среды: компьютерные обучающие тренажеры по классическим методам исследования в офтальмологии, виртуальная химическая лаборатория, моделирование городского района.

Компьютерные обучающие тренажеры внедрены в учебный процесс Тихоокеанского государственного медицинского университета на кафедре офтальмологии и оториноларингологии для обучения студентов лечебного факультета по курсу "офтальмология" и дистанционного обучения врачей.

Программный комплекс использовался в Дальневосточном федеральном университете для выполнения междисциплинарных курсовых проектов.

Разработанные виртуальные среды и программные средства для их разработки и сопровождения доступны для использования на платформе IACPaaS[1] (Intellectual Applications, Control and Platform as a Service) как облачные сервисы.

Положения, выносимые на защиту:

1. Принципы построения профессиональных виртуальных сред как облачных сервисов.

[1] Грибова В.В., Клещев А.С., Крылов Д.А., Москаленко Ф.М., Смагин С.В., Тимченко В.А., Тютюнник М.Б., Шалфеева Е.А. Проект IACPaaS. Комплекс для интеллектуальных систем на основе облачных вычислений // *Искусственный интеллект и принятие решений*. 2011. № 1. С.27-35.

2. Онтология профессиональной виртуальной среды.
3. Декларативная модель профессиональных виртуальных сред.
4. Метод интерпретации декларативной модели.
5. Комплекс программ и технология его применения для построения профессиональных виртуальных облачных сред.

Обоснованность и достоверность полученных результатов обеспечиваются корректным применением использованных в работе методов исследования, а также подтверждаются эффективным практическим применением предложенных в диссертации моделей, методов и программных средств.

Апробация работы. Основные положения диссертации докладывались и обсуждались на следующих международных и российских конференциях и семинарах: Международной конференции "Modern (e-) Learning" (г. Варна, Болгария, 2011, 2013), Международной конференции "Информационно-коммуникационные технологии в образовании "Образование и Виртуальность 2011" (г. Ялта, Украина, 2011), II и III Международной конференции по физиологии и медицине «Высокие технологии, фундаментальные и прикладные исследования в физиологии, фармакологии и медицине», (г. Санкт-Петербург, 2011, 2012), Международной конференции "Новые информационные технологии в образовании" (г. Екатеринбург, 2012), XI Всероссийской научно-технической конференции «Теоретические и прикладные вопросы современных информационных технологий» (г. Улан-Удэ, 2012), Тринадцатой национальной конференции по искусственному интеллекту с международным участием (г. Белгород, 2012), Международной конференции International Conference on Intelligent Computing (г. Хан-шань, Китай, 2012), XIX Международной заочной научно-практической конференции «Технические науки — от теории к практике» (г. Новосибирск, 2013), Конкурсе научных работ молодых ученых и специалистов ИАПУ ДВО РАН (г. Владивосток, 2012), а также на семинарах лаборатории интеллектуальных систем ИАПУ ДВО РАН (2010 – 2013гг.).

Публикация результатов работы. По материалам диссертации опубликовано 18 работ, из них 4 статьи в журналах, рекомендуемых ВАК РФ для опубликования научных результатов.

Структура и объем работы. Диссертационная работа состоит из введения, пяти глав, заключения, списка литературы, включающего 138 наименований, и трех приложений. Работа изложена на 140 страницах машинописного текста, содержит 40 рисунrков и 1 таблицу.

Первая глава содержит обзор литературы. В ней дан анализ современного состояния исследований и разработок в области создания виртуальных сред. Рассмотрены наиболее распространенные инструментальные средства и технологии, применяемые для разработки виртуальных сред.

Во **второй главе** диссертации представлены основные принципы создания и концептуальная архитектура инструментария для разработки и функционирования профессиональных виртуальных сред, программные и информационные компоненты инструментального комплекса. Описана онтология и различные типы моделей профессиональной виртуальной среды.

В **третьей главе** диссертации описывается модель интерпретации виртуальной среды. Рассматривается клиент-серверная архитектура построения программного комплекса, возможные способы хранения данных и обмен данными, процесс перевода декларативной модели из терминов онтологии в программную форму представления.

В **четвертой главе** диссертации описываются методы реализации программного комплекса на основе рассмотренной в предыдущих главах архитектуры, обсуждается выбор технологий и средств реализации: платформы, сред разработки, языков программирования; методы реализации декларативной модели.

В **пятой главе** диссертации описана технология разработки и использования профессиональных виртуальных сред с помощью интегрированного программного комплекса, а также приводятся примеры ПВС, реализованные с его помощью.

В **заключении** формулируются основные результаты, полученные в диссертационной работе.

Личный вклад автора. Все результаты, составляющие основное содержание диссертации, получены автором самостоятельно. В работах [33, 42, 43] автором предложена концептуальная идея по замене разработки виртуальной среды на языке программирования разработкой ее декларативной модели и последующей интерпретацией. В работах [32, 34, 35] автору принадлежит разработка онтологии декларативной модели профессиональных виртуальных сред. В работах [38, 40] автору принадлежит разработка архитектуры программного комплекса. В работах [31, 36, 37] автору принадлежит реализация прототипа компьютерного обучающего тренажера. В работе [65] автору принадлежит разработка и классификация различных типов декларативных моделей. В работах [41, 106, 107] автору принадлежит разработка методов интерпретации виртуальных сред. В работе [39] автором разработан и реализован программный комплекс для создания виртуальных сред, а также технология их создания.

ГЛАВА 1. ВИРТУАЛЬНЫЕ ИНТЕРАКТИВНЫЕ СРЕДЫ И ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ДЛЯ ИХ РАЗРАБОТКИ. ОБЗОР ЛИТЕРАТУРЫ

В данной главе дано определение виртуальных сред, рассмотрены их типы, функции и характеристики, а также основные средства разработки. Особое внимание уделено разработке и сопровождению профессиональных виртуальных сред.

1.1. Виртуальные среды. Общие понятия

Определение виртуальной среды

В литературе встречается множество определений виртуальных сред, виртуальной реальности, виртуальных симуляторов, виртуальных погружений (virtual reality, virtual environment, virtual simulator) обозначающих одну и ту же сущность. Можно отметить, что все источники сходятся на том, что виртуальные среды создаются компьютерными средствами и имеют цель имитации реального мира. В некоторых источниках отмечается возможность виртуальных сред взаимодействовать с пользователем, реагируя на его действия [13, 74, 86, 81-83]. Также говорится о необходимой трехмерности создаваемых виртуальных миров [74] для большего погружения в имитируемую реальность. В некоторых источниках с виртуальными средами непременно упоминаются различные приспособления виртуальной реальности (шлем виртуальной реальности, перчатки виртуальной реальности, и др.) [77, 131]. Иногда [74] обращается внимание на имитацию и отображение на экране движения. В последнее время все больше внимания уделяется использованию виртуальных сред в глобальной сети Интернет [7]. Само появление терминов виртуальных сред связывают [17, 132] с развитием компьютерной техники, так называемой системы «машина-человек», развитием трехмерной визуализации, систем обратной связи.

Возможности применения виртуальных сред широко обсуждается в литературе. В [83] приводятся примеры проведения всевозможных экспериментов с оборудованием, воссоздании технологии отработки различных

нештатных ситуаций. Как отмечено в [97], на необходимость применения систем виртуальной реальности обращают возрастающее внимание как бизнес, так и власти всех уровней. В [132] приводится пример построения новых мультимодальных человеко-компьютерных интерфейсов, с помощью которых разрабатывать тренажеры, симуляторы, обучающие интерактивные системы, цифровые планетарии и т.д.

Области применения виртуальных сред охватывают очень широкие диапазоны. Виртуальные среды с успехом могут применяться и применяются в военных отраслях, в образовании, медицине, развлечениях, бизнесе, инженерии, спорте, в научной визуализации, в архитектуре, в фильмах, в рекламе и т.д [98]. Виртуальные среды дают множество возможностей для эффективного и безопасного обучения и находят применение в различных областях знания [45, 46]. Они способны дать богатые возможности для анализа информации при проектировании сложных систем (в военных отраслях, в инженерии, в науке, в архитектуре). Виртуальные среды способны обеспечить вспомогательную визуальную информацию в дополнение к настоящей реальности (яркий пример – медицина).

Во многих источниках отмечается [21, 77], что первые виртуальные среды стали применяться там, где любая ошибка могла стоить огромных финансовых и человеческих ресурсов, и поэтому требовались средства для преждевременного моделирования ситуаций, проведения безопасных экспериментов. Таким областями применения прежде всего стали военное дело и медицина. В работах [45,46] показано применение виртуальных сред (симуляторов) в обучающей среде уже современного военного вуза. Реальные примеры виртуальных сред в военной области: виртуальный тренажер Raytheon 3D VIRTSIM, в котором, как отмечается в [20], с помощью виртуального обучения тренируются целые подразделения; тренажерный программный комплекс Microsoft Flight Simulator, который дает возможность получить навыки пилотирования в различных метеоусловиях и используется для обучения и даже экзаменования пилотов в различных летных училищах; проект

виртуальной интерактивной среды «Авиация», позволяющий имитировать воздушные полеты в трехмерном пространстве как в ручном, так и в автоматическом режимах со сменой виртуальных камер [67]. Программа FVIEWER, из системы FORAN позволяет интерактивно демонстрировать трехмерную модель судна [75].

Применение виртуальных сред *в медицине* обеспечивает возможность отрабатывать и безопасно проводить различные операции, тренировать моторные навыки, закрепляя их на автоматическом уровне; виртуальные среды применяются в хирургии, в офтальмологии, в терапии, в стоматологии и других областях медицины как для обучения лечению, так и для профилактики и диагностики [104, 119, 124-130]. Пример виртуальной среды для медицины - система HumanSim, в которой отрабатывают различные сценарии взаимодействия врачей и пациентов с помощью трехмерной графической сцены. ЛапСим – виртуальная среда для лапараскопической хирургии и гинекологии. Как следует из [19] данное программное средство предназначается для отработки владения эндохирургическим инструментарием, отработки выполнения эндохирургических вмешательств в абдоминальной хирургии и гинекологии. Симулятор виртуальной среды TestChest предназначен для обучения респираторной терапии [64]. Хирургический тренажер UniSIM, в котором можно проводить тренинги по эндохирургии, лапароскопической и эндоскопической урологии и гинегокологии, артроскопии коленного и плечевого суставов [64]

Во многих источниках также уделяется внимание большому значению применения виртуальных сред в образовании. Виртуальные среды в образовании дают возможность визуализировать лекции и учебные материалы, значительно повышая, таким образом, качество образования. Обучение на тренажерах, как было показано и ранее, имеет огромное значение во многих областях [45-47, 91]. В [3, 4, 80] приводится технология создания в Тверском государственном техническом университете специализированного центра, основными задачами которого являются разработка технологий создания

мультимедиа электронных учебно-методических комплексов и обучающих систем. С помощью компьютерных тренажеров, как отмечено в [30] обучающиеся могут получить практические навыки в виртуальной среде до начало реальной работы, имея возможность «проиграть» различные сценарии возможных ситуаций. Кроме того, что пользователь может виртуально опробовать и проверить свои навыки, он может наглядно увидеть, какие ошибки он допускал и какие действия делал правильно; может посмотреть на себя "со стороны", может посмотреть, как нужно выполнять необходимые действия. Как следует из [50, 51] виртуальные среды в образовании являются логическим развитием более простых обучающих систем (систем тестирования, электронных курсов и др.), предлагая, в отличие от последних, возможность трехмерного отображения и интерактивного взаимодействия с пользователем. В статье [6] описывается пример создания и использования виртуальных тренажеров с целью обучения студентов электротехнике.

Технологии виртуальной реальности находят различные способы применения *в бизнесе*. Это различные виртуальные туры и бизнес-среды, тренировка новых сотрудников, предварительный обзор продуктов, предсказания тенденций развития, испытания продуктов.

Виртуальная реальность уже длительное время находит большое применение *в играх*, благодаря возможности погружения игрока в произвольный искусственный мир (как похожий на реальный, так и полностью фантастический). Особенно отличается от других игр - игра Second Life, представляющая из себя виртуальное социальное пространство, в котором нет типичных для игр целей заработка очков, опыта, «прокачки» персонажа, нет даже сюжетных заданий, однако есть возможность других (социальных) коммуникаций с игроками: общение, путешествия, создание сообществ; видов деятельности, больше приближенных к настоящей реальности: строительство, искусство, виртуальное творчество прямо в пространстве игры [134]. Есть даже целые платформы для разработки таких игровых виртуальных миров, например: *OpenSimulator* (или *OpenSim*), *RealXtend.*

Виртуальные среды могут применяться в различных *социальных сферах, развлекательных сферах, в рекламе*. Могут быть построены *виртуальные музеи, галереи*, организованы виртуальные *туристические маршруты* для ознакомительных целей. Виртуальные среды также могут применяться в различных *демонстрационных* целях.

Пример виртуальной выставки: Проект ExpoVirtual.ru, в котором благодаря глобальной сети Интернет значительно упрощается инфраструктура проведения выставки: нет необходимости в доставке экспонатов, специального оборудования, ну и, главное, нет необходимости в реальном присутствии самих участников выставок, что означает значительное уменьшение финансовых расходов.

Проект «Виртуальная регата», - виртуальная среда, которая представляет собой виртуальный трехмерный остров, окруженный водой, а цель – управлять яхтой при помощи 3D Пойнтера [18].

Проект «Салют» — интерактивная трехмерная презентация с использованием устройства 3D Пойнтер EligoVision. [68]

1.1.2. Типы виртуальных сред

Виртуальные среды можно классифицировать по различным критериям:

- по типу реализации: аппаратные, программные,
- по типу использования: общие и профессиональные,
- по типу взаимодействия с пользователем: интерактивные и неинтерактивные.

Аппаратная и программные виртуальные среды

Аппаратная виртуальная среда – это виртуальная среда, реализованная при помощи различных аппаратных устройств (комнат виртуальной реальности, шлемов, перчаток и других) в дополнении с относительно небольшими программными компонентами. Аппаратная виртуальная среда способна обеспечить максимальное качество погружения пользователя в имитируемую среду. Например, комнаты виртуальной реальности состоят из

компоновки проекционных систем. На стены такой комнаты в реальном времени проецируется 3D стерео-изображение. Пользователь подобной системы может взаимодействовать с виртуальной средой с помощью различных устройств, отслеживающих движения, таких как специальные HMD-шлемы, цифровые перчатки и даже целые костюмы. В работах [22, 23] показываются преимущества использования аппаратных виртуальных сред, в частности применяются устройства: eMagin Z800, проекторы с поляризационными фильтрами, 3 DS-GS Панорама, виртуальная перчатка VHand DG5, трекеры InterSense и другие.

Программная виртуальная среда – это такая среда виртуальной реальности, которой для функционирования достаточно только средств компьютера и никакой дополнительной аппаратуры не требуется. Погружение в виртуальную реальность в таких программных системах обеспечивается за счет применения современной трехмерной компьютерной графики, звука и программных алгоритмов. Благодаря современным информационным и мультимедийным технологиям, компьютерные виртуальные системы способны очень реалистично симулировать поведение реальных объектов. Примеры программных виртуальных сред: виртуальные музеи, виртуальные галереи, системы e-learning, научные визуализации, аналитические бизнес-модели и другие.

По сравнению с аппаратными программные виртуальные среды без дополнительного оборудования не дают возможности взаимодействовать со всеми органами чувств, только на основные – зрение и слух. Однако, для использования программной виртуальной реальности достаточно лишь компьютера, а компьютеры, как известно, в настоящее время есть у абсолютного большинства людей. Благодаря отсутствию необходимости в аппаратных средствах, распространение и применение таких систем также серьезно упрощается, так как вместе с программным обеспечением не требуется распространять еще и сложное оборудование. Обслуживание, доработка и сопровождение п значительно проще, чем аппаратных систем.

В данной работе будут рассматриваться только программные виртуальные среды без каких-либо аппаратных устройств.

Общие и профессиональные виртуальные среды

По типу использования виртуальные среды можно разделить на виртуальные среды общего назначения и профессиональные. Характерными особенностями виртуальных сред общего назначения (их типичным представителем являются компьютерные игры) является заранее определенная логика работы приложения, которая, как правило, в процессе использования мало подвержена изменениям, а также коммерческая направленность реализации, что позволяет привлечь достаточно большие инвестиции (бюджет) к реализации таких сред. Профессиональные виртуальные среды предназначены для решения задач в некоторой предметной области, их основу составляют предметные знания, которые подвержены частым изменениям, а бюджет, за редким исключением очень мал. В настоящее время большинство примеров профессиональных виртуальных сред связано с дистанционными обучающими системами и экспертными системами [73].

Интерактивные и неинтерактивные виртуальные среды

Неинтерактивные среды предназначены для наблюдения, демонстрации каких-либо явлений или процессов, пользователь в таких средах выступает в качестве зрителя; *в интерактивных средах* пользователь может активно взаимодействовать с элементами среды, изменяя ее, может управлять ею и манипулировать по законам виртуального мира [86]. В интерактивных средах пользователь может нажимать на объекты, двигать их, выполнять с ними определенные действия. По сравнению с неинтерактивными виртуальными средами интерактивные виртуальные среды обладают значительно большей степенью погружения в виртуальную реальность за счет обратной связи с пользователем. Примеры виртуальных интерактивных сред: программные компьютерные тренажеры, медицинские системы, виртуальные лаборатории, игры и другие.

1.2. Средства разработки виртуальных сред

В настоящее время существует множество алгоритмов, моделей разработки и сопровождения различных виртуальных сред [11-14, 16, 48, 49, 54, 59, 69, 87]; существуют различные специализированные и универсальные инструментальные средства, пакеты прикладных программ, библиотеки для создания виртуальных сред общего назначения: Дельфин, ToolBook, Lectora, CAVE, WorldToolKit, Avango, Lightning, Juggler [100], Unity3D, Virtools, Alternativa3D, Flare3D и мн. др.

Разработка виртуальной среды может вестись "вручную" как любой другой программной системы. В данном подходе могут применяться любые языки программирования и независимые фреймворки для написания программ, библиотеки и средства: C++, C#, PHP, ActionScript, Visual C++, .Net Framework, Flash CS, Unity, ZendFramework, 3dsmax, Maya и другие [71, 72, 76, 88, 101, 109, 114, 122]. Основные библиотеки и технологии для создания виртуальной среды – это, в первую очередь, библиотеки для работы с компьютерной графикой: OpenGL, DirectX, AlternativaPlatform и другие [5, 85, 105, 122], кроме того, существуют отдельные библиотеки для работы с физикой (PhysX, Havok and ODE) [138] и др.

Разработка может происходить с применением интегрированных инструментальных средств, предназначенных непосредственно для создания виртуальных сред: Unity3D, Vizard Virtual Reality Toolkit, Vrui VR Toolkit, VR Software ViSTA, Neuro VR 2.0, dVise, VR Worx [95, 96, 109, 115]. Программные инструментальные системы для разработки виртуальных сред могут быть проблемно-независимыми (универсальными), охватывающими множество областей виртуальных сред, и могут быть проблемно-ориентированными (специализированными), предназначенными для определенных задач.

Таким образом, существуют различные характеристики инструментальных средств создания виртуальных сред, от которых зависит скорость, удобство, качество и другие показатели разработки.

В большинстве современных библиотек и инструментальных средств разработки виртуальных сред многие важные требования к подобным современным системам реализованы на высоком уровне. Например, качество графики в настоящее время значительно продвинулось вперед по сравнению с прошлыми годами. Появились средства для разработки виртуальных сред для Интернета и мобильных устройств. Например, в 2010 году создана технология Unity3d с поддержкой аппаратной графики в Интернете, в 2011 году появились технологии HTML5 и Molehill (Flash) также с поддержкой Интернета и мобильных устройств. Разработаны многофункциональные мощные IDE для быстрого и качественного кодирования: Visual Studio (.Net Framework), Unity3D и другие.

Однако, существует множество других факторов, влияющих на ценность и применимость инструментального средства и используемых технологий. Далее рассматриваются подробнее различные характеристики инструментальных средств разработки: методы разработки и сопровождения, универсальность, целевая аудитория разработчиков, интерфейс, доступность, технология разработки.

Метод разработки

Большинство инструментальных средств предоставляет возможность разрабатывать собственную логику виртуальной среды. Как правило, это делается с помощью написания необходимых скриптов, управляющих логикой взаимодействия пользователя и объектов среды. На скрипты нет никаких особых стандартов, поэтому в инструментариях используются самые различные языки для написания скриптов. В Vizard Virtual Reality Toolkit скрипты пишутся на Python. В VRSpace используются Java, JavaScript; в Vrui VR Toolkit, ViSTA [95-96] используется C++, в Unity3D используются C#, JavaScript, в Alternativa и Flare используется AS3 (Flash), в dVISE используется собственный текстовый язык описания сцены, объектов и их взаимодействия, скрипты в нем независимы от программы воспроизведения виртуальной реальности, поэтому

для построения виртуальной сцены по новым или измененным скриптам не требуется перекомпиляция или переписывание кода программы.

Скрипты позволяют описывать произвольную интерактивность виртуальной сцены. Однако, различаются стили и подходы в их использовании. В некоторых задачах более уместно применение "императивного подхода", где последовательно кодируется вся функциональность. В некоторых задачах более предпочтительным является "декларативный" подход, в котором на достаточно высоком (абстрактном) уровне описываются независимые элементы и их функциональность. В большинстве инструментальных средств используется императивный подход. И лишь в немногих из рассмотренных – декларативный (например, в dVISE).

Инструментальные средства, которые не поддерживают программирование в разрабатываемых виртуальных средах, как правило, носят узко-специализированный характер. В VR Worx [117] все задачи и сценарии заранее известны: например, просмотр товара с разных сторон, панорамные обзоры и они, в принципе, могут обходиться без интерактивного взаимодействия с пользователем, которому вполне достаточно просто "увидеть" объект.

Метод сопровождения

Хорошо известно, что в современных программных продуктах на разработку программы тратится значительно меньше времени и усилий, чем на сопровождение программы. Таким образом, важным фактором для программы виртуальной среды является простота и удобство ее сопровождения.

В удобство сопровождения входят различные составляющие:

1) Необходимость перекомпиляции исходных кодов программы. Это является требованием для большинства рассмотренных средств и связано с "императивным" стилем программирования.

2) Технология обновления программы у пользователей, которые уже применяют ее. Традиционные десктопные программы, которыми являются все рассмотренные средства разработки виртуальных сред, требуют либо полной

переустановки программного обеспечения, либо скачивания и установки специальных патчей. Программы, которые способны работать как облачные сервисы, вообще не устанавливаются на компьютере пользователя, поэтому разработчикам достаточно обновить ее только на сервере, пользователи же автоматически получат обновленную версию программы. Однако, облачных инструментальных средств разработки виртуальных сред не было найдено.

Универсальность

Инструментальные средства могут быть предназначены для разработки виртуальных сред в произвольных и конкретных предметных областях.

Проблемно-независимый инструментарий

Проблемно-независимые инструментальные средства обычно имеют значительно больше возможностей и гибкости, чем проблемно-ориентированные, однако, сложность разработки в универсальных средствах также значительно выше. В универсальных системах для разработки виртуальной среды необходимо привлечение высоко-квалифицированных программистов, которым потребуется, используя обобщенные структуры и возможности, создать специализированную систему [77, 99, 100, 116].

Для управления и сопровождения виртуальной среды требуются высококвалифицированные программисты, регулярно появляется необходимость дописывать и переделывать код. В случае проблемно-независимых систем проблема стоит еще более остро, так как возможности универсальных инструментальных средств не предназначены для какой-то конкретной задачи.

Проблемно-ориентированный инструментарий

Проблемно-ориентированные системы предназначены для решения специфических задач в узкой предметной области. Такие системы сложно применять для разработки виртуальных сред во многих областях. В случае необходимости использования такой инструментальной системы для схожей предметной области или задачи, могут появиться значительные трудности.

Проблемно-ориентированные инструментальные системы сложно интегрировать с другими системами или технологиями [77, 99, 100, 117].

Проблемно-ориентированные инструментальные системы являются в большинстве своем монолитными программными системами, не рассчитанными на какие-либо внешние дополнения. [115, 116, 132]. Это означает часто значительную трудность или невозможность добавить дополнительные функции в такое инструментальное средство. Для сопровождения проблемно-ориентированных систем требуются высоко квалифицированные узкие специалисты. Такие специалисты должны хорошо знать как технологии компьютерной виртуальной реальности, так и узкие направления той предметной области или задачи, для которых данное инструментальное средство создавалось. Это накладывает достаточно серьезные трудности и ограничения.

Разделение работы между специалистами

Важным фактором в процессе создания виртуальной среды является то, какие специалисты и каким образом принимают участие в разработке, как взаимодействуют между собой и как зависят друг от друга.

Несомненно, что во многих узких областях всей полнотой информации о необходимой среде обладают только эксперты этих областей. Однако, эти эксперты, как правило, не являются программистами, инженерами, компьютерными специалистами, и поэтому не в состоянии обычно самостоятельно вести разработку. Во многих средствах создания виртуальных сред необходимо писать программный код даже для создания простейшей сцены. Кроме того, разработчикам необходимо уметь пользоваться различными редакторами: как редакторами из средства разработки, так и другими дополнительными редакторами (например, графическими 2d и 3d редакторами для подготовки требуемых для сцены текстур, моделей и т.д.). В литературе отмечается, что в разработке многих виртуальных сред должны принимать участие различные специалисты. [73, 83] При этом, чем сильнее будет разделение задач и обязанностей между различными специалистами, тем лучше

будет общая производительность разработки. Однако, абсолютное большинство средств разработки приложений виртуальных сред, рассмотренных автором, не предлагает удобных способов вести параллельную разработку различным специалистам.

В одних средствах разработку способен вести только программист (имеются только редакторы скриптов, или консольные редакторы), эксперты и дизайнеры же должны консультировать программиста. Это наблюдается в следующих средствах разработки ВС: VRSpace, Vrui VR Toolkit, ViSTA, dVISE, Alternativa 3D

В других средствах, с мощными интегрированными редакторами, можно провести разделение на виды работ: дизайнерам можно отдельно редактировать представление сцены, модели объектов и т.д. Однако, в данных средствах модели (графические образы объектов на сцене), скрипты, управляющие ими действиями с ними, логика сценария (выполненная программистом), которой должно подчиняться все происходящее на сцене, - все это очень сильно связано друг с другом и часто не может быть отделено друг от друга. Примеры таких средств: Vizard Virtual Reality Toolkit, Unity, Flare3D.

В некоторых узкоспециализированных средствах имеется возможность разработки логики программы без участия программистов. Однако, в таких средствах, сильно ограничены возможности по интерактивности и функциональным возможностям виртуальной среды; эксперты применяют готовые шаблоны с заранее написанными сценариями, при этом сами не имеют возможности добавить что-то важное, чего нет в стандартных шаблонах. Другими словами, в таких средствах разработки эксперты способны только вносить неизменные данные о виртуальной среде. Примеры таких средств: VR Worx, Neuro VR.

Интерфейс

Качество интерфейса инструментальной системы означает полноту и удобство предлагаемых средств для всех групп разработчиков.

В программных инструментальных средствах могут быть традиционные (Vizard Virtual Reality Toolkit, VRSpace, Vrui VR Toolkit, ViSTA, Unity3D) и веб-интерфейсы (VR Worx).

Основной вид интерфейсных элементов инструментальной системы при разработке виртуальных сред – это редакторы. Инструментальные средства по созданию виртуальных сред предоставляют различные редакторы: редакторы кода, редакторы сцен, редакторы отдельных моделей, интегрированные редакторы, узко-специализированные редакторы, редакторы физики, звука и т.д.. Наличие редакторов зависит от целей, преследуемых инструментальным средством, и направления конечных приложений.

Некоторые рассматриваемые средства создания виртуальных сред представлюет из себя библиотеки, поэтому не предполагают встроенных редакторов. Например, HTML5 – это просто технология создания трехмерных сцен в браузере, к ней не предполагаются никакие редакторы, но данная технология может быть использована в применяющих эту технологию средствах. Разработчики библиотеки Alternativa3D предоставляют дополнительно несложный редактор для обработки отдельных моделей – объектов сцены.

Существуют также средства разработки виртуальных сред, которые не являются библиотеками, но, либо не предоставляют вообще никаких редакторов, либо предлагают к использованию сложный для применения консольный редактор без графического интерфейса, который, как правило, предназначается для редактирования скриптов. Примеры таких средтв разработки виртуальных сред: Vrui VR Toolkit, ViSTA – достаточно мощные универсальные инструментальные средства создания виртуальных сред, но без каких-либо редакторов. В dVISE имеется редактор для создания скриптов.

Если рассматриваемое инструментальное средство нацелено только на создание демонстрационных показов товаров, то там, будет редактор, позволяющий только загружать модели, правильно их размещать для показа, накладывать какие-либо специальные эффекты. Это, с одной стороны,

упрощает разработку конкретных узко-специализированных задач, но с другой стороны и сильно ограничивает ее. В VR Worx предоставляется три отдельных специализированных редактора: 1) для создания панорам, 2) для создания демонстрационных показов объектов-товаров, 3) для создания виртуальных туров-галерей.

В некоторых средствах разработки виртуальных сред имеется мощный универсальный интегрированный редактор, однако, сильно увеличивается сложность его использования, что может снизить скорость разработки и сопровождения.

В Neuro VR редактор позволяет обычным пользователя, не экспертам, редактировать виртуальную сцену. В редакторе имеется стандартный набор заранее подготовленных виртуальных сцен и отдельная библиотека объектов, которые можно добавлять на сцену, изменять, убирать и т.п.

В Unity мощный интегрированный редактор, позволяющий в едином месте редактировать (проектировать, моделировать) виртуальную сцену, писать для нее код и тестировать, запуская локальный проигрыватель.

Во Flare3D мощный редактор для подготовки виртуальных сцен и моделей. Однако, он не является интегрированным, скрипты необходимо писать в других инструментальных средствах.

В Vizard Virtual Reality также имеется интегрированный редактор. Основные особенности: визуальный редактор свойств объектов сцены, отладчик скриптов, возможность запускать скрипты во время выполнения симуляции, автодополнение кода, контроль версий.

Доступность

В настоящее время поддержка Интернета является одним из важнейших факторов, так как именно Интернет дает возможность многим пользователям попробовать и применять программный продукт без лишних усилий и затрат. Интернет обеспечивает простоту доступа и использования для пользователей, распространение и простую модификацию программ для разработчиков. При этом возможность работы в Интернет подразумевает не просто видео, которое

можно посмотреть в браузере (как в Vizard Virtual Reality Toolkit и VR Worx), а полноценную интерактивную среду, способную работать на веб-страничке.

Для традиционных настольных компьютерных систем разработка трехмерной виртуальной реальности предполагает использование преимущественно движков и библиотек, основанных на программных интерфейсах, как DirectX, OpenGL [5, 85, 105]. Данные технологии позволяют использовать всю мощь компьютера и создавать гипер-реалистичную компьютерную графику и другие эффекты. Однако, несмотря на высокий уровень качества и возможностей, предоставляемых традиционными средствами, они не предназначены для Интернета. Это такие десктопные инструментальные системы, как: Vizard Virtual Reality Toolkit, Vrui VR Toolkit, ViSTA, dVISE. Программы, получаемые с помощью таких систем, сохраняются в исполняемом формате EXE и не предназначены для запуска в Интернете [95, 96, 115, 116].

Средства и технологии, поддерживающие Интернет, также можно разделить на разные категории.

Есть средства, работающие на "стандартных" Интернет-технологиях, таких как VRML, HTML5 [1, 62, 84, 89, 123]. VRML уже считается устаревшим, так как не поддерживает аппаратного ускорения графики, которая в итоге эмулируется процессором компьютера и имеет очень низкое качество. VRML используется, например, в системе VRSpace. На смену VRML пришел HTML5 - современная технология разработки виртуальных сред для Интернета, которая использует аппаратное ускорение видео-карты компьютера и работает на программном интерфейсе OpenGL. Для HTML5 не требуются никакие специальные плагины для браузера. Однако, для данной технологии в настоящее время разработаны только фреймворки и редакторы, предназначенные для создания сайтов с красивым мультимедиа наполнением, каких-либо мощных средств разработки виртуальных сред не обнаружено. Связано это с тем, что сама технология еще не достаточно развита, в ней отсутствуют многие важные для виртуальных сред элементы [123].

Есть средства, работающие в Интернете через специальные плагины, устанавливаемые в браузер пользователя. В большинстве рассмотренных средств используются свои собственные плагины, которые, соответственно, мало распространены и поэтому являются препятствием для пользователей из-за необходимости их установки. В VRSpace используется собственный клиентский модуль, у Unity3D – свой собственный плагин [109]. Некоторые средства используют стандартные плагины, т.е. плагины, разработанные другими разработчиками для общих целей. Например, VR Worx [117] использует формат выходного файла QuickTime, в Alternativa3d, Flare3D используется Flash-формат swf. Большое преимущество стандартных плагинов (таких как Flash) заключается в их хорошей распространенности у пользователей, что не становится преградой при использовании, так как нет необходимости что-то устанавливать. Кроме того, у плагинов обычно больше возможностей, чем у стандартных технологий VRML, HTML5 – лучше качество графики, быстрее скорость работы, больше доступных функций.

Важно, чтобы разработанная программа работала у максимального количества пользователей в независимости от установленной операционной системы на компьютере или используемого браузера.

В предыдущие годы средства разработки виртуальных сред делались, как правило, без учета кроссплатформенности, были ориентированы на какую-то определенную аппаратуру и программное обеспечение. В настоящее время ограничения на кроссплатформенность также возникают у систем, которые рассчитаны на работу со сложным аппаратным обеспечением для более глубокого погружения в виртуальную среду (такие как шлемы, перчатки и т.п.), а также трудности с кроссплатформенностью бывают у узко-специализированных систем, выполненных, например, по определенному заказу для какой-то задачи на некотором предприятии. В остальных случаях современные технологии и средства хорошо поддерживают задачу кроссплатформенности (Alternativa3d, Flare3d, Unity3d, VRSpace, Neuro VR, VR Worx).

Технологии разработки

Одним из наиболее важных факторов, определяющих ценность средства разработки виртуальных сред, является технология создания готового продукта, от которой напрямую зависит скорость и удобство разработки. Несомненно, скорость разработки может сильно различаться в зависимости от сложности создаваемого приложения.

В общем случае процесс создания виртуальной среды состоит из нескольких основных этапов: разрабатываются 3D-модели объектов виртуального мира [71], для большинства моделей объектов описываются сценарии их поведения и возможного изменения отображения в виртуальной среде; затем из разработанных объектов формируется виртуальная среда (виртуальное окружение) – определяется положение объектов относительно друг друга, их размер, повороты и др. необходимые атрибуты; отдельной трудоемкой задачей является описание возможных сценариев как влияния объектов друг на друга, так и изменения виртуального мира при воздействии на него пользователя; функционал некоторых интерактивных сред требует включения оценки действий пользователя при его взаимодействии с виртуальным миром.

Технология разработки профессиональных виртуальных сред для большинства рассмотренных инструментальных систем является «традиционной» и заключается в следующем:

1. Эксперты предметной области описывают задачу и логику виртуальной среды. Делают они это на бумаге или в текстовых редакторах (например, MS Word).
2. Эксперты объясняют программистам и дизайнерам все детали необходимой виртуальной среды.
3. Программисты программируют логику, описанную экспертами.
4. Дизайнеры создают графическое представление элементов виртуальной среды.

5. Программисты добавляют специальный код, загружающий и применяющий все созданные дизайнерами графические элементы.

Шаги 2 и 3 повторяются в цикле, пока эксперты не подтвердят правильность логики, реализованной программистами.

Шаги 2 и 4 повторяются в цикле, пока эксперты не подтвердят правильного графического представления элементов предметной области.

Шаги 4 и 5 повторяются до тех пор, пока графическое представление не будет установлено в правильное соответствие с логикой виртуальной среды.

На рис. 1.2.1 представлена общая технология разработки виртуальных средствами общего назначения.

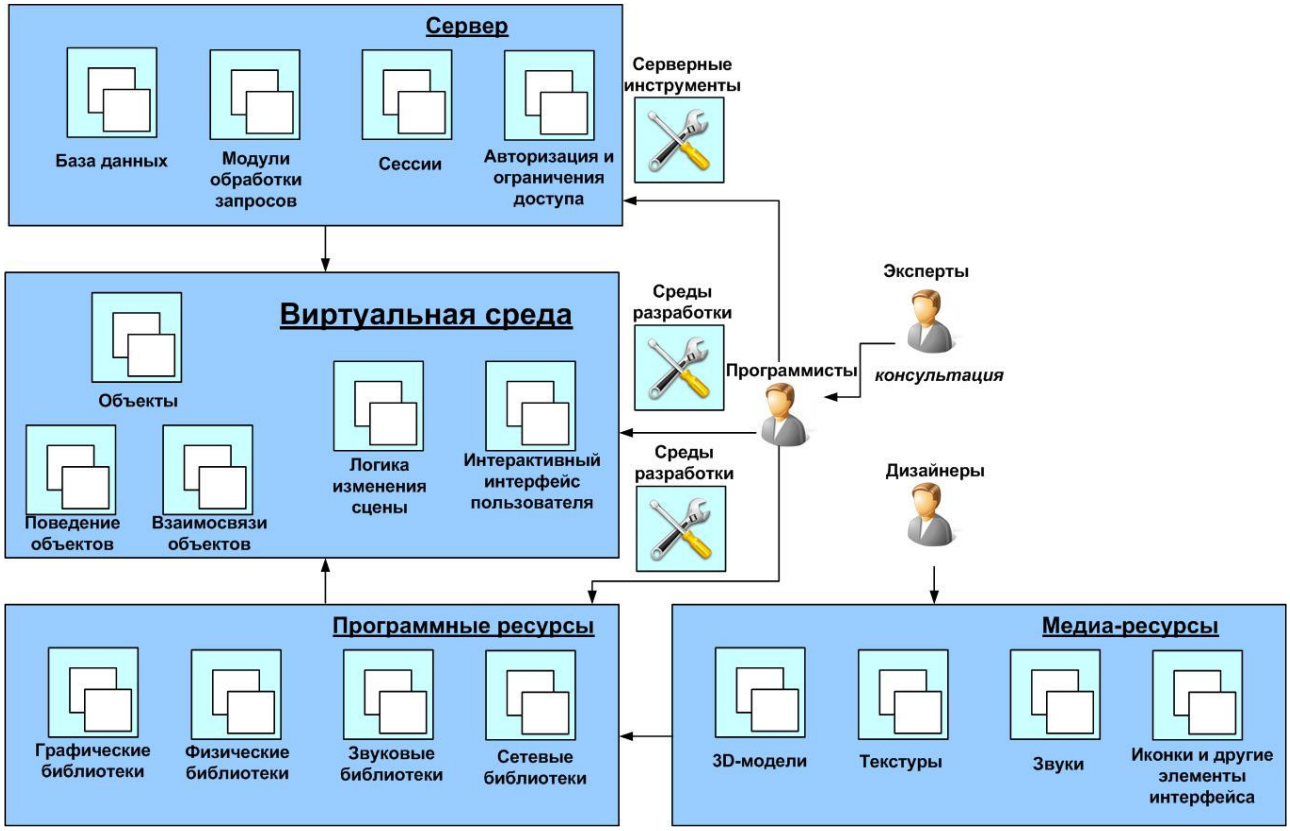


Рис. 1.2.1. Стандартная технология разработки виртуальных сред

1.3. Выводы из обзора существующих средств

1. Из проведенного обзора следует, что в современном мире виртуальные среды являются очень актуальными, востребованными и перспективными технологиями в очень многих областях применения. Несмотря на достаточно количество разработанных виртуальных сред общего назначения (особенно

игр), количество разработанных на настоящий момент профессиональных виртуальных сред остается очень небольшим.

2. Обзор существующих средств и технологий разработки виртуальных сред показал, что для создания виртуальных сред общего назначения существует множество различных инструментальных комплексов и технологий, удовлетворяющих по качеству графики, интерфейса, интегрированной среде разработки и т.д. Однако в настоящее время неизвестны специализированные средства для разработки именно профессиональных виртуальных сред. Использование инструментов общего назначения для создания профессиональных виртуальных сред делает процесс их разработки и особенно сопровождения чрезмерно трудоемким и дорогим.

3. Анализ существующих средств создания профессиональных виртуальных сред показывает, что для эффективной их разработки необходима, во-первых, интеграция существующих технологий виртуальной реальности, облачных технологий, и, во-вторых, современных методов, обеспечивающих упрощение и автоматизацию процесса разработки и сопровождения с привлечением специалистов предметной области.

Таким образом, на основе представленного выше обзора средств разработки виртуальных сред и сделанного на его основе анализа можно сделать вывод об актуальности разработки нового программного инструментального комплекса для создания профессиональных виртуальных сред как облачных сервисов.

ГЛАВА 2. АРХИТЕКТУРА ИНСТРУМЕНТАРИЯ ДЛЯ РАЗРАБОТКИ ПРОФЕССИОНАЛЬНЫХ ВИРТУАЛЬНЫХ СРЕД. ОНТОЛОГИЯ И МОДЕЛЬ ПРОФЕССИОНАЛЬНОЙ ВИРТУАЛЬНОЙ СРЕДЫ

В данной главе представлены основные принципы создания и концептуальная архитектура инструментария для разработки и функционирования профессиональных виртуальных сред, программные и информационные компоненты инструментального комплекса. Описана онтология виртуального мира и различные типы моделей профессиональной виртуальной среды.

2.1. Основные принципы создания инструментария для разработки и функционирования профессиональных виртуальных сред

Основными проблемами, отмеченными в обзоре литературы, является трудоемкость проектирования, реализации и особенно сопровождения профессиональных виртуальных интерактивных сред, невозможность включения в процесс разработки экспертов предметной области, а также обеспечение доступности профессиональных виртуальных сред широкому кругу заинтересованных пользователей. Для устранения указанных недостатков предлагаются следующие основные принципы создания инструментария [33, 42, 43].

1) Автоматизация процесса разработки профессиональной виртуальной среды (ПВС)

Для реализации данного принципа предлагается заменить проектирование, реализацию и сопровождение ПВС на языке программирования проектированием и сопровождением декларативной модели ПВС с последующей ее интерпретацией, сведя к минимуму этап кодирования на языке программирования.

Модель ПВС должна позволять описать на декларативном языке ПВС различного назначения для произвольных предметных областей. Обеспечить возможность для включения в декларативную модель императивных модулей с целью расширения потенциального функционала инструментария в процессе его жизненного цикла.

Сведение к минимуму трудоемкого этапа кодирования направлено на снижение общей трудоемкости разработки и особенно сопровождения, которое по оценкам специалистов занимает до 70 % общей трудоемкости в жизненном цикле программной системы.

Использование интерпретации модели вместо ее компиляции также направлено на упрощение сопровождения [24, 102, 103, 111]. В случае использования интерпретатора все изменения в модели сразу можно просмотреть и в случае необходимости отладить запуском интерпретатора модели; при использовании компилятора сначала требуется скомпилировать модифицированный код, запустить приложение и в случае совершения ошибки повторить компиляцию и запуск.

2) Включение в процесс разработки специалистов разного профиля, экспертов предметной области, дизайнеров, программистов и разделение работы между ними.

Для реализации данного принципа предлагается создать онтологию виртуальной среды, выделить в ней логический и презентационный уровни, в терминах которой разработчики ПВС будут разрабатывать и сопровождать модель конкретной ПВС. Создание моделей осуществлять через структурный и графический редакторы, управляемые онтологией.

Использование онтологии [2, 15, 29, 52, 53, 70, 90, 93] профессиональной виртуальной среды, описывающей систему понятий виртуального мира, связей между ними и ограничения целостности направлено на упрощение создания модели ПВС, поскольку разработчикам не надо изучать какой-либо формализм для описания модели (особенно это важно для экспертов предметной области, не знакомых с языками программирования и технологией разработки программных систем); создание модели производится в терминах онтологии. Использование редакторов, управляемых онтологией ПВС, также направлено на снижение трудоемкости разработки. Выделение в онтологии двух уровней – логического и презентационного направлено на

разделение работ между разработчиками – экспертами предметной области и дизайнерами.

3) Использование как средства разработки ПВС, так и готового приложения как интернет-сервисов.

Использование Интернета и технологии облачных вычислений в общем случае, как отмечено в ряде источников [7, 21], дает новый уровень гибкости для управления и сопровождения программных средств, а также значительно расширяет аудиторию пользователей.

Разработка ПВС и использование готового приложения через web-браузер (что является требованием технологии облачных вычислений) не требует сложной установки программного средства на компьютере конечного пользователя или разработчика, не предъявляет никаких дополнительных условий к операционной системе, оперативной памяти и др. техническим требованиям их компьютеров. Очень важно использование данной технологии для обеспечения жизнеспособности готового приложения, поскольку в процессе всего жизненного цикла модель ПВС доступна для сопровождения. Практически только эта технология и позволяет обеспечить модифицирование модели в процессе ее использования. Удаленное использование позволит не только расширить аудиторию пользователей средства, но также и привлекать разработчиков независимо от их географического расположения.

2.2. Концептуальная архитектура инструментального комплекса

В соответствии с основными принципами создания инструментария для разработки и функционирования профессиональных виртуальных сред сформирована общая концепция архитектуры программного комплекса (см. рис. 2.2.1).

Декларативные модель создается разработчиками программного комплекса – экспертами предметной области и дизайнерами. Модель формируется на основе онтологии виртуальной среды с помощью структурного и графического редакторов. В моделях могут быть использованы различные мультимедиа-данные. Затем модели интерпретируются уже в процессе работы с

ней конечными пользователями. Все это работает как облачный сервис, как для разработчиков, так и для пользователей.

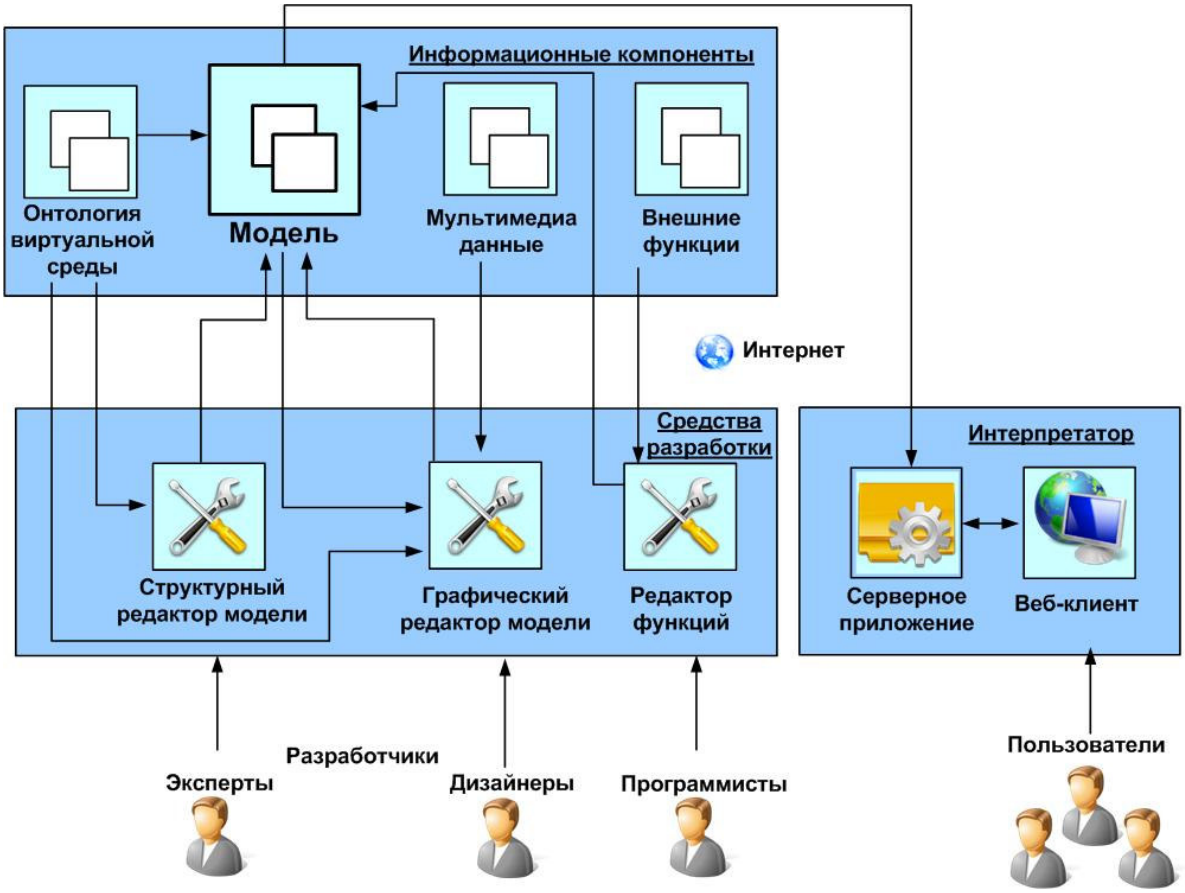


Рис. 2.2.1. Концептуальная архитектура программного комплекса

Программный комплекс (система) состоит из нескольких компонентов, которые можно разделить на два класса: информационные и программные.

Информационными компонентами являются онтология виртуальной среды, декларативные модели профессиональной виртуальной среды, построенные на основе онтологий; мультимедиа-данные (картинки, 3d-модели, необходимые для построения и правильного отображения виртуальной среды и т.д.);

Программными компонентами являются:

• Структурный редактор модели – редактор для создания и изменения всех компонентов модели на основе онтологии.

• Графический редактор модели – интерактивное средство редактирования виртуальной среды ПВС на основе декларативной модели. Графический

редактор модели предназначен для дизайнера и отображает виртуальную среду так, как будет видеть ее пользователь. Вся внесенная или измененная с помощью этого редактора информация также является декларативной.

• Интерпретатор (серверное приложение) – программа для обработки и анализа декларативной модели и изменения в соответствии с ней виртуальной среды.

• Интерпретатор (веб-клиент) – программа, отображающая в браузере пользователю виртуальную среду, полученную в результате интерпретации модели.

Для описания системы понятий виртуального мира, связей между ними и ограничения целостности разработана онтология профессиональной виртуальной среды. Реализация онтологии представлена в Приложении 2.

2.3. Онтология виртуальной среды

Онтология описывается с использованием теоретико-множественной нотации. Онтология представляет собой тройку: объекты, действия и сценарий [31, 32, 34, 35]:

Онтология = <*Объекты, Действия, Сценарий*>,

где *Объекты* – объекты виртуальной среды, *Действия* – действия, которые можно выполнить с объектами виртуальной сцены, *Сценарий* - сценарий проверки действий пользователя

Далее применяются следующие обозначения:

Перед тем, как описывать классы онтологии, введем *Типы данных*, используемые в ней.

Типы данных

Множество типов данных состоит из множеств простых и составных типов данных:

Типы данных = <*Простые типы, Составные типы*>,

где *Простые типы* – множество простых типов данных, *Составные типы* - множество составных типов данных.

Множество простых типов данных включает в себя: целочисленный тип, вещественный тип, строковый тип, логический тип.

Простые типы = {Целое число, Вещественное число, Строка, Логическое значение}, где:

- *Целое число* – числовой тип, представляющий класс целых чисел.
- *Вещественное число* – числовой тип, представляющий класс вещественных чисел.
- *Строка* – строка.
- *Логическое значение* – логический тип, принимающий одно из двух значений: ложь (0) или истина (1).

Множество составных типов данных включает в себя следующие элементы: множество, вектор, интервал целых чисел, интервал вещественных чисел, вычисляемое значение:

Составные типы = {Множество, Вектор, Интервал целых чисел, Интервал вещественных чисел, Вычисляемое значение}, где:

- *Множество*. Включает в себя пары из ключей и значений:

Множество = {<Имя$_i$, Значение$_i$,> | Имя$_i$ ∈ Строка, Значение∈ Типы данных}, 1<= i <= n, где n – количество элементов в множестве, *Имя$_i$* – название элемента множества, *Значение$_i$* – значение элемента множества.

- *Вектор* – упорядоченное множество значений одного типа. *Вектор = <Значение$_i$ | Значение$_i$ ∈ Типы данных >*, 1<= i <= количество элементов
- *Интервал целых чисел*. Описание интервала - пара, задающая нижнюю и верхнюю границу интервала целых чисел *Интервал целых чисел = <Нижняя граница, Верхняя граница>,*
- где *Нижняя граница ∈ Целое число,*
- *Верхняя граница ∈ Целое число,*
- *Верхняя граница > Нижняя граница.*
- *Интервал вещественных чисел.* Описание интервала - пара, задающая нижнюю и верхнюю границу интервала вещественных чисел.

- *Интервал вещественных чисел = <Нижняя граница, Верхняя граница>,*
- *Нижняя граница ∈ Вещественное число,*
- *Верхняя граница ∈ Вещественное число,*
- *Верхняя граница > Нижняя граница.*
- *Вычисляемое значение* – значение, вычисляемое с помощью вызова функции. Включает в себя имя вызываемой функции, входные параметры, передаваемые этой функции и тип возвращаемого значения.
- *Вычисляемое значение = <Имя функции, Параметры, Возвращаемое значение>* где
- *Имя функции ∈ Все функции, Все функции – это множество всех функций, доступных в системе*
- $Параметры = <Имя_i, Тип_i | Имя_i \in Строка, Тип_i \in Строка, >_{i=1}^{n}$
- *Возвращаемое значение ∈ Типы данных.*

Объекты

Неформальное описание

Объекты виртуальной среды включают описание множества всех объектов, находятся в виртуальной среде, разделенных на следующие классы: простой (неизменяемый) объект, изменяемый объект, составной объект, таблица. Помимо основных объектов виртуальная сцена содержит служебные объекты: источники света и камеру.

Простой объект – это объект, обладающий набором атрибутов, которые не изменяются в процессе воспроизведения сцены. Простой объект можно назвать "базовым" в силу того, что от него наследуются все остальные типы объектов.

Изменяемый объект – это объект, который является расширением простого объекта, т.е. обладает всеми атрибутами, которые есть у простого объекта, а также включает множество изменяемых атрибутов, которые могут быть самостоятельными или объединены в логические группы – состояния. Каждое состояние определяется совокупностью значений атрибутов, которые они могут принимать в ответ на действия, происходящие в виртуальной среде.

Составной объект – это объект, который наследуется от изменяемого объекта, обладает всеми его атрибутами и, кроме этого, может содержать дочерние объекты, образуя иерархию. Для образования иерархической структуры у родительского объекта должны быть дополнительные атрибуты - "указатели" на дочерние объекты. Эти атрибуты могут быть изменяемыми (находиться на уровне состояний). Если в некотором состоянии у объекта есть этот атрибут, то объект, указанный в нем, становится дочерним по отношению к данному объекту. В разных состояниях могут быть указаны различные наборы дочерних объектов, в некоторых состояниях их может не быть, что будет означать, что у объекта нет дочерних объектов в этом состоянии. Дочерние объекты могут быть объектами любых типов, их вложенность не ограничена. Все дочерние объекты (или подобъекты) обязательно имеют состояния, соответствующие состояниям родительского объекта, т.е. состояния с одинаковыми именами, а также и свои уникальные состояния, которых нет у родительского объекта. Такая иерархическая связь обеспечивает управление всем сложным объектом, как единым целым. При изменении состояния родительского объекта, соответствующие состояния принимают и дочерние объекты. Примером составного объекта является объект «пациент», его дочерними объектами являются: «рука правая», «рука левая», «глаз правый», «глаз левый» и т.п.

Таблица – это объект, предназначенный для описания, хранения и управления набором (множеством) подобных или одинаковых объектов. Таблица наследуется от изменяемого объекта. Специфическими атрибутами данного объекта являются элементы таблицы, содержащие описание ячеек таблицы.

Источник света – это служебный объект, не имеющий собственного отображения, и служащий для указания в пространстве сцены точки и направления излучения света, а также визуальных характеристик света. Источник света наследуется от изменяемого объекта, т.к. его атрибуты могут быть динамически изменены.

Камера – это служебный объект, который также как и источник света не имеет собственного отображения, и служащий для указания в пространстве сцены точки, в которой находится пользователь. Объект камера наследуется от простого объекта.

В общем случае описание объекта любого класса состоит из **имени** (уникального идентификатора объекта), текстового **описания** объекта и **множества атрибутов**. Атрибуты объектов классифицируются по разным критериям.

По уровню определения:

- *Атрибуты логического уровня (логические атрибуты)* – атрибуты, которые задает эксперт предметной области, они определяют характеристики объектов предметной области.
- *Атрибуты презентационного уровня (презентационные атрибуты)* – атрибуты, которые устанавливаются дизайнером, они задают характеристики отображения объектов в виртуальной среде. Презентационными атрибутами являются: модель (3d-образ объекта, геометрическая сетка объекта), текстура (это растровое изображение, накладываемое на одну из поверхностей объекта для придания ей цвета, окраски или иллюзии рельефа), координаты (три вещественных числа x, y, z, которые определяют положение объекта на сцене в мировой системе координат), углы вращения (три вещественных числа, углы вращения вокруг осей x, y, z, от нуля до 360 градусов), коэффициенты масштабирования (три вещественных числа, множители по осям x, y, z, которые необходимы для изменения размеров объекта), анимация (изменение образа объекта, циклично повторяющееся через одинаковые промежутки времени), переход (анимационный) - однократное изменение образа объекта в течение некоторого промежутка времени.

По изменяемости:

- *неизменяемые атрибуты* – это атрибуты объекта, значения которых не изменяются в процессе воспроизведения сцены;

- *изменяемые атрибуты* – это атрибуты объекта, значения которых меняются в зависимости от действий, произведенных над объектом, команд, которые выполняет объект (одушевленный), а также изменения атрибутов других объектов.

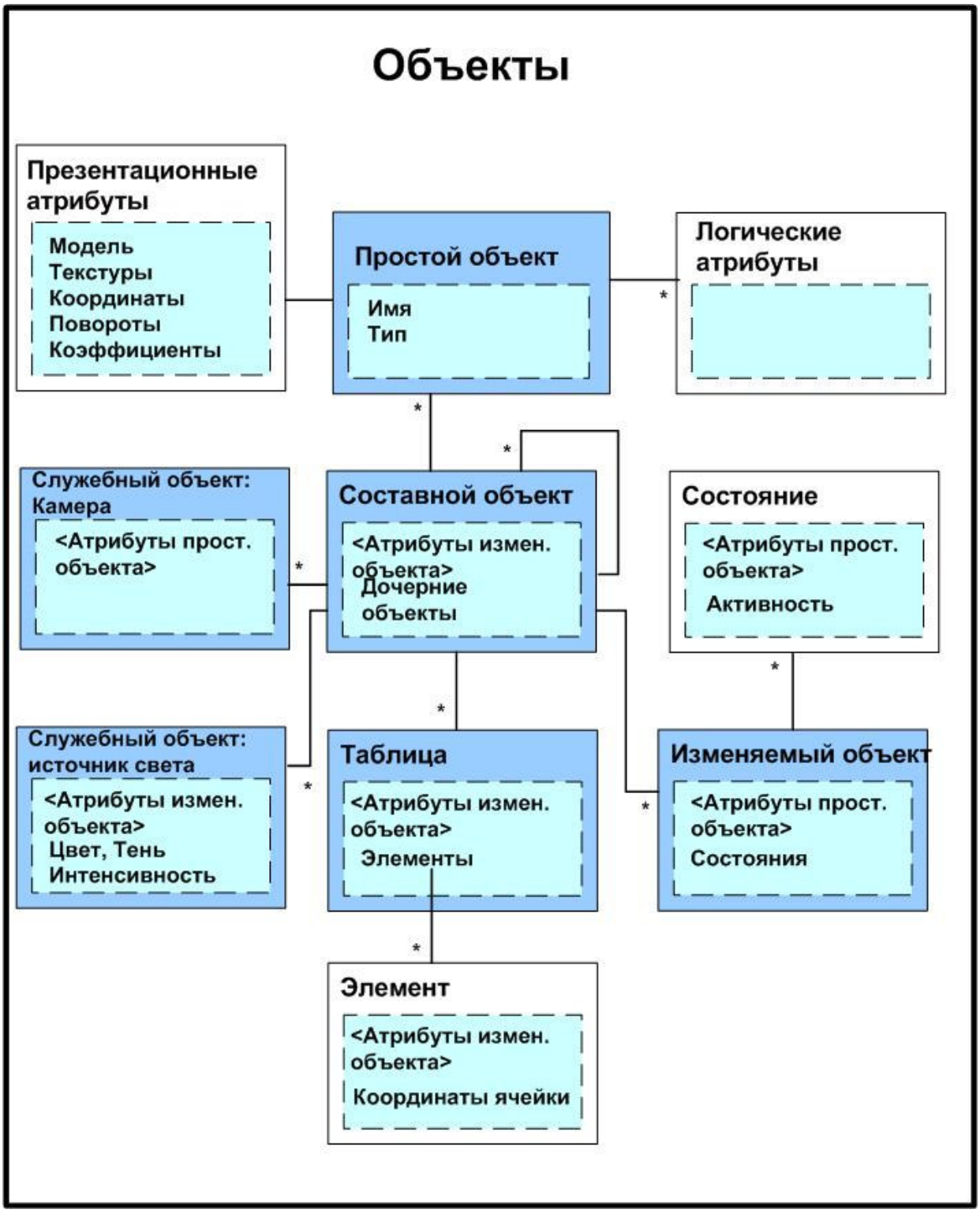


Рис. 2.3.1. Онтология объектов

По критерию необходимости:

- *обязательные атрибуты* – это атрибуты, которые должны присутствовать обязательно, либо при описании объекта, либо при описании его состояний;
- *необязательные атрибуты* – это атрибуты, которые могут отсутствовать.

На рис. 2.3.1 представлено схематическое изображение онтологии объектов.

Формальное описание

Объекты = {Простой объект, Изменяемый объект, Составной объект, Таблица, Источник света, Камера}$_i$ 0<= i <= количество объектов

Простой объект = <*Имя объекта, Описание, Логические атрибуты, Презентационные атрибуты* >

Имя объекта∈ Строка, имя объекта, его идентификатор, по которому к нему можно получить доступ

Описание ∈ Строка

Логические атрибуты = {*Логический атрибут*}$_i$, 0<= i <= количество логических атрибутов

Логический атрибут = <*Имя* ∈ Строка, *Значение*∈ Типы данных>

Презентационные атрибуты = <*Модель, Текстуры, Координаты, Повороты, Масштабы, Анимация* >

Модель ∈ Строка,

Модель – это 3d-модель, которая будет использоваться для создания трехмерного образа объекта. Данная строка может содержать либо название 3d-модели, имеющейся в библиотеке стандартных 3d-моделей, либо путь к внешней 3d-модели.

Текстуры – это текстуры, используемые для покрытия 3d-модели.

Текстуры = {*Текстура*}$_i$, 0<i<количество текстур. Каждая текстура – это либо название текстуры, имеющейся в стандартной библиотеке текстур, либо пути к внешним текстурам.

Текстура ∈ Строка

Координаты – это три вещественных числа, определяющих положение объекта в пространстве.

Координаты = <КоординатаX ∈ Вещественное число, КоординатаY ∈ Вещественное число, КоординатаZ ∈ Вещественное число>

Повороты – это три вещественных числа, определяющих поворот объекта относительно соответствующих трех локальных осей

Повороты = <ПоворотX ∈ Вещественное число, ПоворотY ∈ Вещественное число, ПоворотZ ∈ Вещественное число>

Масштабы – это три вещественных числа, определяющих масштабирование объекта относительно соответствующих трех локальных осей

Масштабы = <МасштабX ∈ Вещественное число, МасштабY ∈ Вещественное число, МасштабZ ∈ Вещественное число>

Анимация ∈ Строка,

Анимация – это название анимации, которая должна быть воспроизведена для данного объекта

Изменяемый объект = <*Имя объекта, Описание, Логические атрибуты, Презентационные атрибуты, Множество состояний изменяемого объекта, Начальное состояние изменяемого объекта* >

Множество состояний изменяемого объекта = {*Состояние изменяемого объекта*}$_i$, 1<= i <= количество состояний объекта

Состояние — это множество атрибутов объекта с заданными значениями.

Состояние изменяемого объекта = <*Имя, Логические атрибуты, Презентационные атрибуты состояния изменяемого объекта* >

Презентационные атрибуты состояния изменяемого объекта = <*Модель, Текстуры, Координаты, Повороты, Масштабы, Анимация, Переход анимации*>

Начальное состояние изменяемого объекта – это одно из множества состояний изменяемого объекта. *Начальное состояние изменяемого объекта* ∈ Множество состояний изменяемого объекта.

Переход анимации = <*Анимация перехода, Время, Интерполяция*>

Анимация перехода ∈ Строка,

Анимация перехода – это название анимации, которая должна быть воспроизведена во время перехода из предыдущего состояния в текущее. Анимация перехода выполняется только в момент перехода из одного состояния в другое, атрибут "Анимация" может быть задан в состоянии, в том числе и в начальном состоянии.

Время ∈ Вещественное число,

Время – это время, которое будет затрачено для перехода из предыдущего состояния в текущее.

Интерполяция ∈ Логическое значение

Интерполяция – это флаг, который определяет, нужно ли интерполировать значения презентационных атрибутов объекта во время перехода из предыдущего состояния в текущее состояние.

Составной объект = <*Имя объекта, Описание, Логические атрибуты, Презентационные атрибуты, Множество состояний составного объекта, Начальное состояние изменяемого объекта, Дочерние объекты*>

Дочерние объекты = {Простой объект, Изменяемый объект, Составной объект, Таблица, Источник света, Камера}$_i$ 0 <= i <= количество дочерних объектов.

Множество состояний составного объекта = {Состояние составного объекта}$_i$, 1<= i <= количество состояний объекта

Состояние составного объекта = <Имя, Логические атрибуты, Презентационные атрибуты состояния изменяемого объекта, Дочерние объекты >

Начальное состояние составного объекта – это одно из множества состояний изменяемого объекта.

Начальное состояние составного объекта ∈Множество состояний составного объекта.

Таблица = <*Имя объекта, Описание, Логические атрибуты, Презентационные атрибуты, Множество состояний изменяемого объекта, Элементы*>

Элементы = {*Элемент*}$_i$ 1<= i <= количество элементов.

Элемент = <*Презентационные атрибуты элемента, Логические атрибуты*>

Презентационные атрибуты элемента – это атрибуты, описывающие положение ячейки элемента на плоскости объекта. Для этого используются координаты левого верхнего угла элемента (x1, y1) и правого нижнего угла элемента (x2, y2).

Презентационные атрибуты элемента = < $x_1 \in$ Вещественное число, $y_1 \in$ Вещественное число, $x_2 \in$ Вещественное число, $y_y \in$ Вещественное число>

Источник света = <*Имя объекта, Описание, Презентационные атрибуты источника света, Множество состояний изменяемого объекта, Цвет, Сила, Тень, Тип*>

Презентационные атрибуты источника света = <Координаты, Повороты, Масштабы>

Цвет определяет цвет испускаемого источником света

Цвет $\in$ Вещественное число.

Сила определяет интенсивность испускаемого источником света

Сила $\in$ Вещественное число.

Тень определяет, будет ли данный источник света отбрасывать тени от объектов

Тень $\in$ Логическое значение

Тип определяет вид источника света: точечный или направленный

Тип = {"Точечный", "Направленный"}

Камера = <*Имя объекта, Описание, Презентационные атрибуты источника света*>

Онтологические соглашения

1) На сцене может быть не более одной камеры

2) У одного объекта не может быть двух одноименных атрибутов.
3) Один и тот же объект не может одновременно являться дочерним и родительским по отношению к другому объекту
4) Все имена объектов должны быть различны.
5) Множество состояний изменяемого объекта не должно быть пустым.
6) Все состояния одного объекта должны быть различны
7) Для каждого изменяемого объекта должно быть задано начальное состояние, являющимся одним из множества состояний объекта.
8) Для каждого составного объекта в каждом состоянии должны быть указаны только те дочерние объекты, которые являются дочерними для данного объекта в данном состоянии. Дочерние объекты, указанные в предыдущем состоянии составного объекта и не указанные в текущем состоянии, уже не будут являться дочерними объектами для данного составного объекта.

Действия

Неформальное описание

Действия в онтологии виртуальной среды описывают все действия, которые можно производить с объектами виртуальной среды. Выделяются следующие основные виды действий:

- *командные действия* - это действия, которые пользователь совершает над объектами вне пространства виртуальной среды (с помощью элементов интерфейса, голоса);
- *интерактивные действия* – это действия, которые пользователь совершает над объектами, находясь в пространстве виртуальной среды.
- *простые действия* – это действия, которые пользователь не выполняет непосредственно, а которые выполняются виртуальной средой на основе параметра другого действия

Действия могут быть проверены с помощью *входных параметров*. Входные параметры определяют, выполнены ли необходимые условия для обработки данного действия.

Результаты выполнения действий могут быть связаны с изменением виртуальной среды или получением информации. Изменение виртуальной среды предполагает изменение объектов виртуальной сцены путем изменения их состояний. Таким образом, можно выделить два основных вида действий *по назначению*:

1) *действия, изменяющие объекты виртуальной среды*;

2) *действия, возвращающие информацию пользователю*.

1. Действия, изменяющие объекты виртуальной среды. В описании указываются имена объектов и состояния, которые должны быть изменены. Кроме того, перед изменением состояния объекта, проверяются параметры возможности изменения объекта; объект изменяет состояние только в случае, если все указанные параметры прошли успешную проверку.

2. Действия, возвращающие информацию пользователю, - это действия-функции, результатом выполнения которых является текстовая информация. Информация формируется в результате обработки действия на основе множества *значений оценок* и *множества значений параметров*, определяемых для атрибута *получение оценки*. Эти два множества значений и параметров оценок используются для описания соответствий между оценками результатов, атрибутами объектов и множествами их возможных значений. Множество значений оценок представляет собой множество константных значений возможных оценок, которые затем используются во множестве значений параметров. Множество оценок параметров включает соответственно набор *оценок параметров*. *Оценка параметров* – это набор сравниваемых параметров и одного из заданных значений оценки. Если все сравниваемые параметры истинны, то соответствующая оценка становится результирующей. Каждый *параметр* включает: *ожидаемое значение* – константное значение, *значение* – ссылка на существующий атрибут объекта или параметр действия и *операцию сравнения*, определяющую метод сравнения *ожидаемого значения* и *значения*. Таким образом, в данных множествах значений оценок и оценок параметров определяются экспертные знания – соотношения между различными

вариантами значений, оценок и т.п., используемых в соответствующих методиках предметной области.

Рис. 2.3.2. Структура действий

Результат действия состоит из двух частей:

- *результат в виртуальной среде* – это результат, который отображается пользователю в виртуальной среде после завершения действия.

- *результат для сценария* – это результат, который передается в сценарий для анализа состояния виртуальной среды и не отображается пользователю.

На рис. *2.3.2 представлена структура* действий, отображающая основные виды действий и их атрибуты, параметры.

Формальное описание

Действия = *{*Действие$_i$ | Действие$_i$ ∈ Действие*}*, 0<= i <= количество действий

Действие = <*Имя действия, Описание, Способы выполнения, Входные параметры, Изменение состояния объектов, Получение оценки, Параметры обработки, Сообщение*>

Имя действия – уникальный идентификатор действия

Имя действия ∈ Строка

Описание ∈ Строка

Способы выполнения включают в себя различные способы выполнения действия. *Способы выполнения* – это множество, т.е. одновременно может быть задано несколько способов выполнения.

Способы выполнения ∈ {<*Интерактивные, Командное, Простое*}>

Интерактивные = { *Интерактивное* }$_i$, 0<= i <= количество интерактивных способов выполнения

Интерактивное = <*Объект, Событие*>

Объект -один из описанных ранее объектов.

Объект ∈ Объекты

Событие выбирается одной из доступных констант (нажатие мышкой, наведение мышкой).

Событие ∈ {*MOUSECLICK, MOUSEOVER*}

Входные параметры определяют параметры, проверку которых необходимо осуществить перед выполнением действия.

Входные параметры = *{Входной параметр*$_i$ |*Входной параметр*$_i$ ∈ *Параметр}*, 1<= i <= количество входных параметров

Параметр = <*Значение, Ожидаемое значение, Операция сравнения*>

Ожидаемое значение – это определенная экспертом константа, ожидаемая в качестве значения параметра. С этой константой будет осуществляться проверка значения параметра.

Значение – это значение произвольного атрибута объекта или действия. Атрибут в форме ссылки определяет, что значение не является константой, а меняется вместе со значением атрибута, на который указывает.

Операция – это одна из констант, определяющих требуемую операцию сравнения.

Значение∈ Типы данных

Ожидаемое значение∈ Типы данных

Операция сравнения ∈ {=, !=, >, <, >=, <=}

Изменения состояния объектов определяют множество объектов, у которых должны быть изменены состояния.

Изменение состояния объектов = { Изменение состояния объекта }$_i$, 0 <= i <= количество изменений состояний объектов

Изменение состояния объекта – это изменение состояние объекта при выполнении условий, заданных в параметре. Объект задается ссылкой на один из существующих объектов, у которого есть множество состояний, это либо Изменяемый объект, либо Составной объект, либо Источник света. Параметр аналогичен входному параметру с ожидаемым значением, проверяемым значением и операцией сравнения.

Изменение состояния объекта = <{ *Состояние изменяемого объекта, Состояние составного объекта, Состояние источника света* }, *Параметр*>

Получение оценки = <*Значения оценок, Множество оценок параметров*>

Значения оценок – это множество заранее определенных значений оценок, которые используются при дальнейшем анализе параметров.

Значения оценок = { *Значение оценки* ∈ Строка }$_i$, 0 <= i <= количество значений оценок.

Множество оценок параметров = { *Оценка параметров* }$_i$, 0 <= i <= количество оценок параметров

Оценка параметров = <*Параметры*, *Значение оценки*>

Параметры = { *Параметр* }$_i$, 0 <= i <= количество параметров

Значение оценки ∈ Значения оценки

Параметры обработки – это набор параметров, которые должны обрабатываться внешним образом, специфическими функциями, невстроенными в систему.

Параметры обработки = { *Параметр обработки* }$_i$, 0 <= i <= количество параметров обработки

Параметр обработки = <*Функция обработки* ∈ Вычисляемое значение, *Значение* ∈ Типы данных>

Сообщение ∈ Строка

2.4. Сценарий

Неформальное описание

Общий принцип работы пользователя с объектами виртуальной среды основан на том, что пользователь может совершить все действия, доступные в некотором состоянии виртуальной среды, независимо от того, являются они правильными или не являются. Сценарий предназначен для описания последовательности действий, которые должен выполнить пользователь для получения результата, определяемого обучающим заданием. Таким образом, сценарий в онтологии виртуальной среды задается только в том случае, если виртуальная среда является формой представления обучающего задания, в результате выполнения которого пользователь должен получить результат, правильно ли выполнено обучающее задание. И в случае ошибочного выполнения получить объяснение – какие действия были ошибочными, и какие действия ожидались от него на каждом шаге выполнения задания.

С одной стороны, сценарий представляет собой граф, в вершинах которого находятся действия, а дуги являются переходами от одних действий к другим. С другой стороны, сценарий – это последовательность логических этапов, где каждый этап представляет собой объединенное в группу множество вершин. *Этапы* могут иметь параметры, уточняющие особенности его

выполнения. *Вершины* графа представляют собой ожидаемые от пользователя действия, связанные *дугами* (*переходами*).

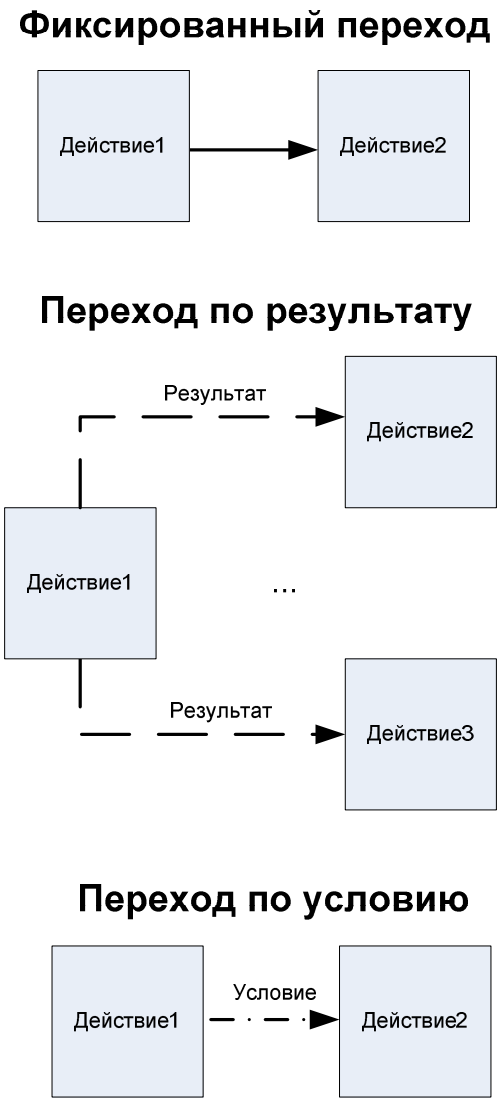


Рис. 2.4.1. Виды переходов между действиями

Переходы могут быть:

- *фиксированные* – это постоянные переходы, которые не имеют зависимости от результата;
- *по результату* – это переходы, которые зависят от результата действия, имеют несколько взаимоисключающих вариантов переходов;
- *с условием* – это переходы, которые выполняются только при наличии выполненного условия, независимо от конкретных результатов предыдущего действия.

На рис. 2.4.1 представлены все описанные выше виды переходов между действиями.

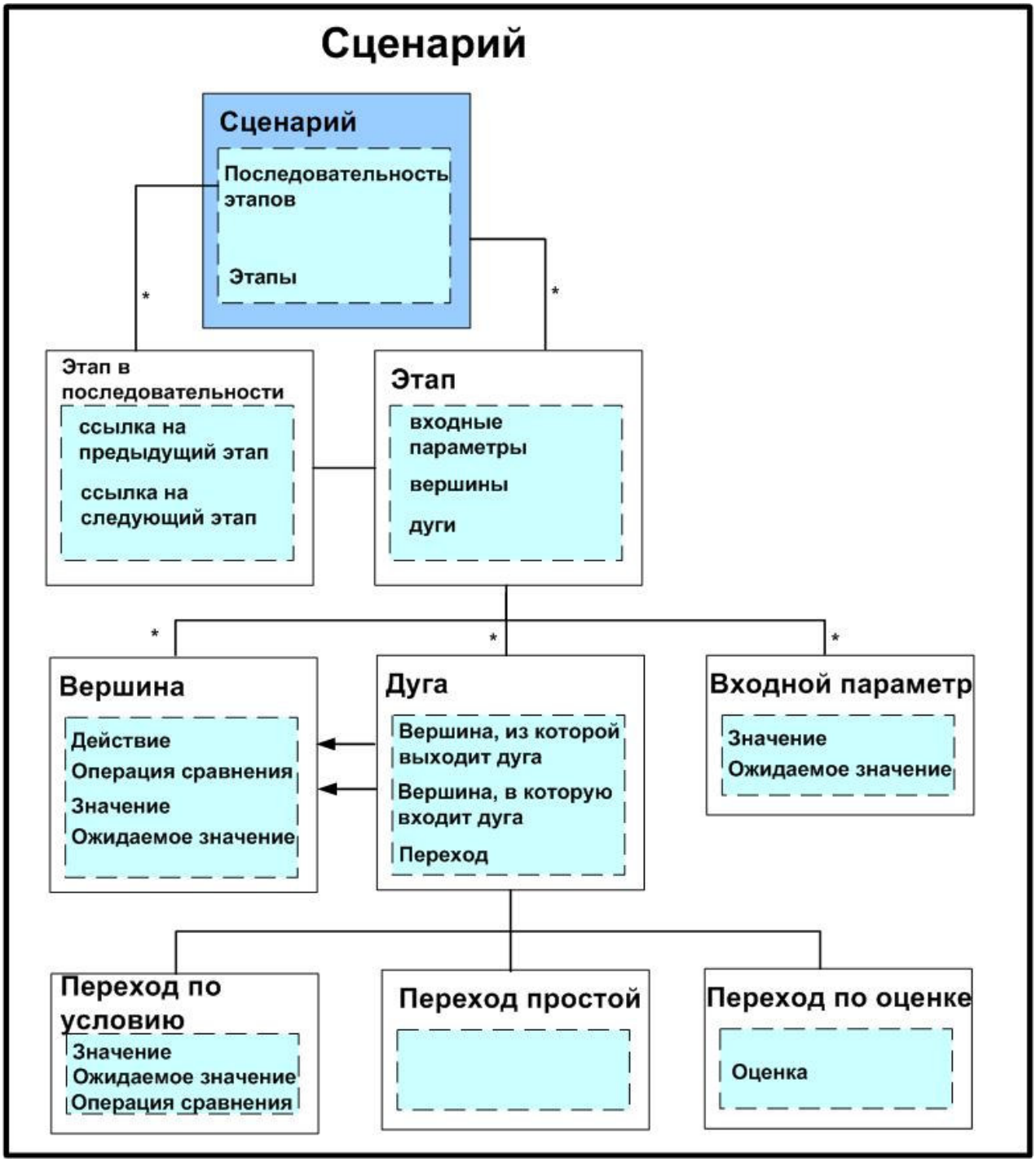


Рис. 2.4.2. Структура сценария

На рис. 2.4.2 представлена структура сценария.

Формальное описание

Сценарий = <Имя сценария, Этапы, Последовательность этапов>

Имя сценария∈ Строка

Этапы = { *Этап* }$_i$, 0 <= i <= количество этапов

Этап = <*Входные параметры, Вершины, Дуги, Начальная вершина, Конечная вершина*>

Вершины = {*Вершина*$_i$∈ *Вершина* }, 0<i<= количество вершин

Дуги = {*Дуга*$_i$∈ *Дуга*}, 0<i<= количество дуг

Начальная вершина ∈ *Вершина*

Конечная вершина ∈ *Вершина*

Вершина = <*Действие, Параметр*>

Параметры = {*Параметр*$_i$ ∈ *Параметр*}, 0<i<= количество параметров действия

Дуга = <*Вершина из которой выходит дуга, Вершина в которую входит дуга, Переход*>

Вершина из которой выходит дуга, ∈ *Вершина*

Вершина в которую входит дуга ∈ *Вершина*

Переход ∈ {*Простой, По оценке, По условию*}

По оценке = { *Оценка* }$_i$, 0 <= i <= количество этапов

По условию = <*Значение, Ожидаемое значение, Операция сравнения*>

Уровни онтологии

В соответствии с принципом разделения разработки между различными специалистами, экспертами и дизайнерами, в онтологии выделяется два уровня: логический (для экспертов) и презентационный (для дизайнеров).

Важным моментом в создании онтологии является принципиальное разделение так называемых «уровней» разработки: одну часть декларативной модели по этой онтологии должны будут формировать эксперты, а другую часть – дизайнеры, после того, как эксперты сформировали логическое описание модели

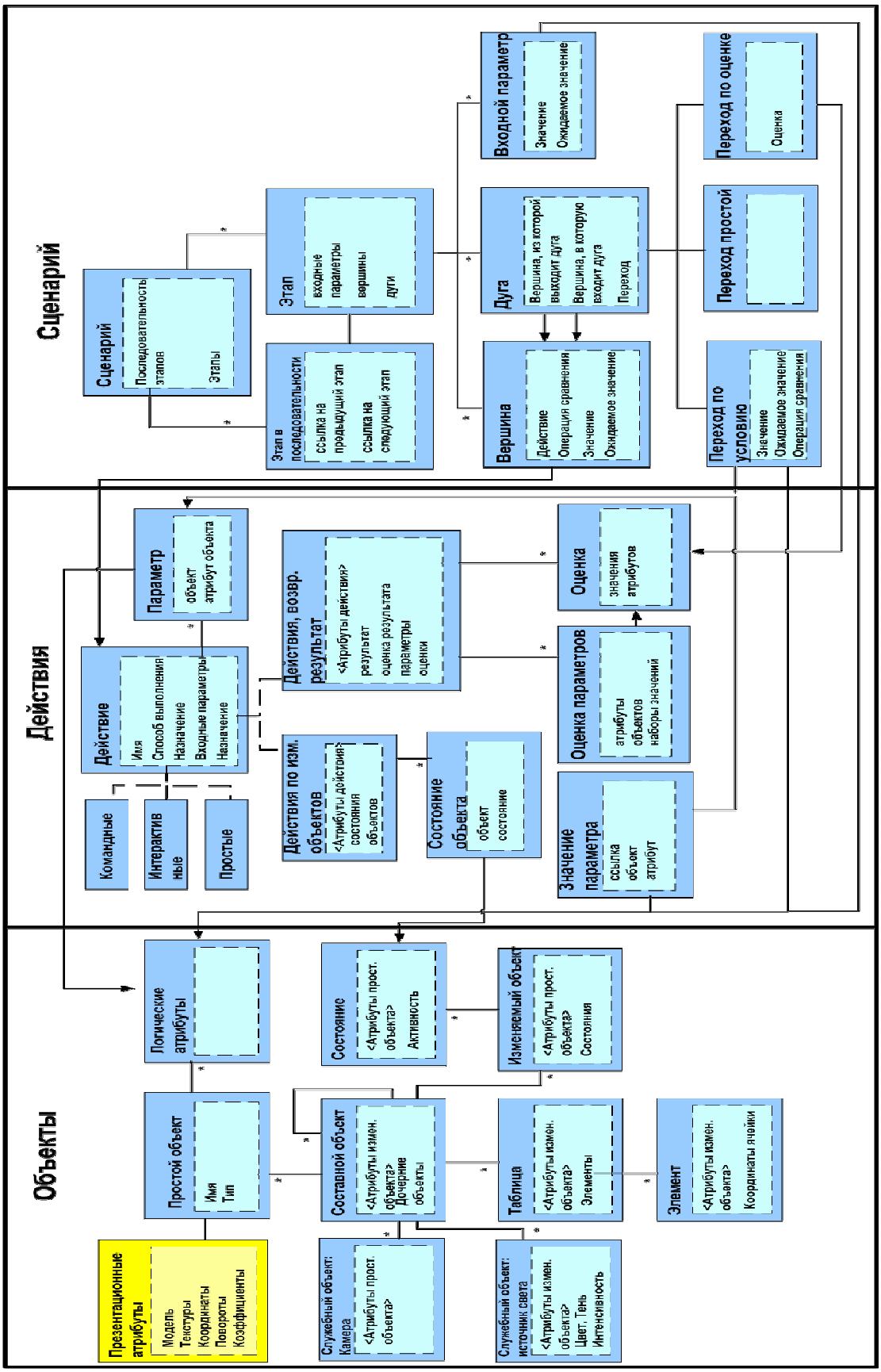


Рис. 2.4.3. Уровни онтологии виртуальной среды

Основная часть декларативной модели виртуальной среды формируется экспертами. Дизайнеры через визуальное средство разработки, дополняют логическое описание модели ПВС его визуальным представлением которое поможет сформировать «дизайнерский» уровень декларативной модели.

Таким образом, выделяется два уровня онтологии (см. рис. 2.4.3):

Логический уровень (логические атрибуты) – на этом уровне находятся атрибуты, которые задает эксперт предметной области, они определяют характеристики объектов предметной области.

Презентационный уровень (презентационные атрибуты) – на этом уровне находится атрибуты, которые устанавливаются дизайнером, они задают характеристики отображения объектов в виртуальной среде.

2.5. Декларативная модель

Ниже представлены типы [42, 65] декларативных моделей профессиональных виртуальных сред, их неформальное и формальное описание, ограничения целостности.

Типы моделей виртуальных сред

Виртуальные среды можно разбить на следующие основные типы:

- неинтерактивные статичные виртуальные среды – среды со статическим виртуальным наполнением, без возможности интерактивного взаимодействия для пользователя. Такие среды предназначаются для демонстрации каких-либо предметов, статичных сцен. Неинтерактивные статичные среды могут быть использованы для теоретического обучения. Для описания проекта такой виртуальной среды достаточно лишь онтологии и описания объектов сцены
- неинтерактивные анимационные виртуальные среды – это среды, подобные предыдущему типу статичных сред с разницей в том, что некоторые объекты сцены могут быть анимированы и изображать, например, какой-нибудь процесс.
- интерактивные виртуальные среды без контроля – среды, позволяющие пользователю взаимодействовать с объектами сцены, получая от них обратную реакцию. Интерактивные виртуальные среды предназначаются для

практического обследования и изучения виртуального мира рассматриваемой области; позволяют исследовать виртуальную сцену, проводить на ней различные виды экспериментов. Объекты и их реакция на действия пользователя могут меняться в процессе работы программы. Для описания проекта интерактивной виртуальной среды необходимо описать объекты и действия в соответствии с описанными ранее онтологиями.

- интерактивные виртуальные среды с контролем – среды с интерактивным взаимодействием по заранее заданному сценарию. В таких средах программа может следить за выполнением действий пользователя, помогать ему, объясняя ошибки, отображая заданные и полученные результаты.

Схемы представленных типов моделей виртуальных сред представлены на рис. 2.5.1.

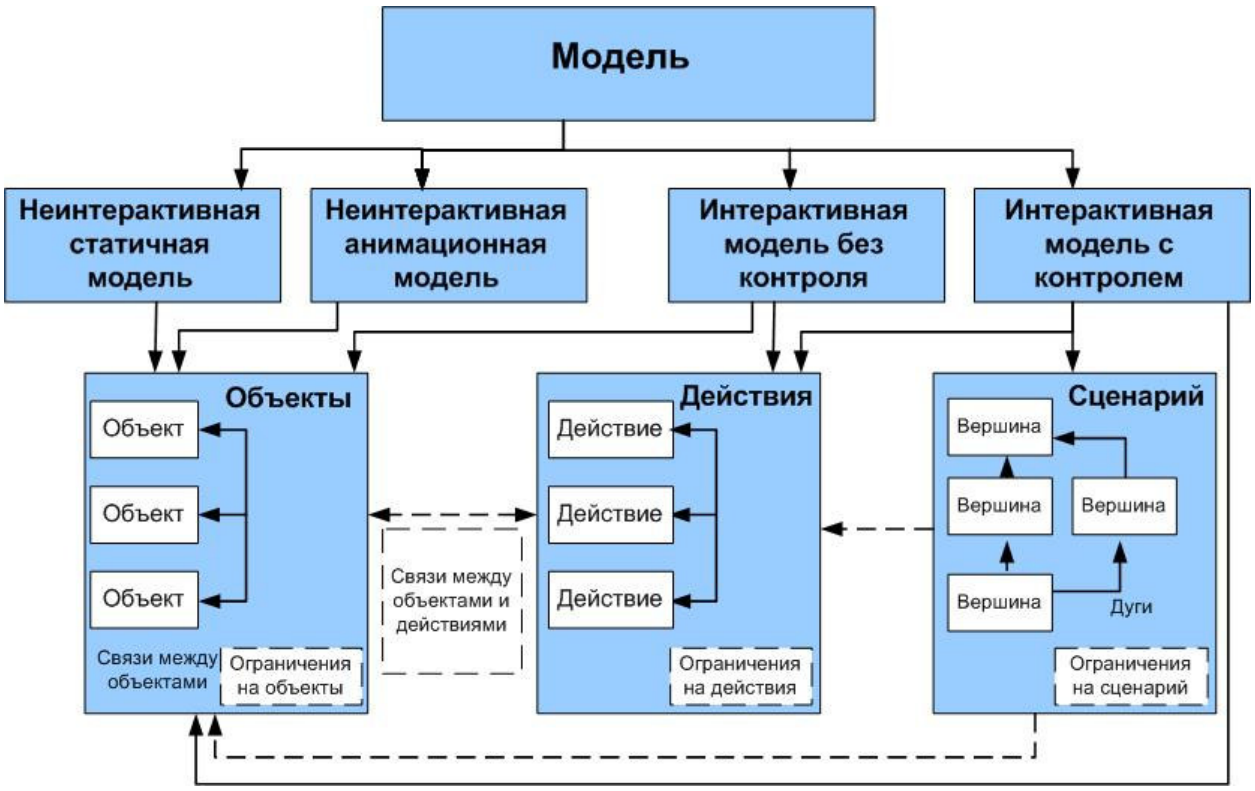


Рис. 2.5.1. Типы моделей виртуальных сред

В общем случае декларативная модель виртуальной среды представляет собой конкретизацию онтологии и описывает конкретные ее характеристики: уточняет множество объектов виртуальной среды, их атрибутов и значений, действия, которые могут быть произведены в виртуальной среде, а также задает

сценарии выполнения действий пользователя. Таким образом, на основе одной онтологии можно построить бесконечное множество различных виртуальных сред. Для каждого типа виртуальной среды описывается отдельная декларативная модель.

Декларативная модель неинтерактивной статичной среды

Декларативные модели этого типа представляют собой множество объектов.

Декларативная модель = <Объекты среды>

Объекты среды = {Объект$_1$,Объект$_2$,…,Объект$_N$},

где *Объект$_i$* = Простой объект | Изменяемый объект | Составной объект | Таблица | Источник света | Камера , *N* – это количество объектов виртуальной среды

Декларативная модель неинтерактивной анимационной среды

Декларативные модели этого типа аналогичному типу представляют собой множество объектов среды.

Декларативная модель = <Объекты среды>

Объекты среды = {Объект$_1$,Объект$_2$,…,Объект$_N$},

где *Объект$_i$* = Простой объект | Изменяемый объект | Составной объект | Таблица | Источник света | Камера , *N* – это количество объектов виртуальной среды

Декларативная модель интерактивной среды без контроля

Декларативные модели этого типа представляют собой множество объектов и множество действий, на которые накладываются ограничения целостности.

Декларативная модель = <Объекты среды, Действия среды >

Объекты среды – это множество объектов среды,

Объекты среды = {Объект$_1$,Объект$_2$,…,Объект$_N$},

где *Объект$_i$* = Простой объект | Изменяемый объект | Составной объект | Таблица | Источник света | Камера , *N* – это количество объектов виртуальной

среды *Действия среды* – это множество действий, которые пользователь может производить в виртуальной среде

Действия среды = {Действие$_1$, Действие$_2$,…,Действие$_N$},

где *Действие$_i$ = Действие, N – это количество действий*

Ограничения целостности:

Имена всех объектов должны быть различны.

Имена всех действий должны быть различны.

Если в действии задан атрибут изменения состояний объектов, то объект, входящий во множество объектов, заданных в этом атрибуте, должен входить во множество объектов сцены.

Для любого объекта Объект$_i$, $\subseteq$ Изменение состояний объекта $_i$ $\subseteq$ Изменение состояний объектов $\subseteq$ Действие$_k$ Объект$_i$ $\subseteq$ Объекты сцены, 1 <= i <= количество изменений состояний объектов в действии, 1 <= k <= количество действий сцены.

Если в действии задан атрибут изменения объектов, то состояние объекта, указанное для объекта, входящего во множество объектов, заданных в этом атрибуте, должно входить во множество состояний этого объекта.

Для любого объекта Объект$_i$, $\subseteq$ Изменение состояний объекта $_i$ $\subseteq$ Изменение состояний объектов $\subseteq$ Действие$_k$ и соответствующего состояния Состояние$_i$ $\subseteq$ Изменение состояний объекта $_i$ $\subseteq$ Изменение состояний объектов $\subseteq$ Действие$_k$, Состояние$_i$ $\subseteq$ Множество состояний $\subseteq$ Объект$_i$

Если тип действия по назначению - "действия, возвращающие результат пользователю", то у него должен быть задан атрибут "получение оценки", "значения оценок", "множество оценок параметров". Множество "значения оценок" должно быть непустым. Множество "множество оценок параметров" должно быть непустым. Любая оценка параметров, входящая во множество оценок параметров, должна содержать ссылку либо на логический атрибут одного из существующих объектов сцены, либо на один из параметров обработки агентов, заданных для действия.

Для любого объекта $Объект_i$, $\subseteq$ $Значение_i$ $\subseteq$ *Входные параметры* $\subseteq$ $Действие_k$ $Объект_i$ $\subseteq$ Объекты сцены, 1 <= i <= количество используемых входных параметров, 1 <= k <= количество действий сцены,

Если в действии задан атрибут "входные параметры", то атрибут, указанный в параметре для объекта, входящего во множество объектов, заданных в этом атрибуте, должен входить во множество атрибутов этого объекта.

Для любого объекта $Объект_i$, $\subseteq$ $Значение_i$ $\subseteq$ *Входные параметры* $\subseteq$ $Действие_k$ и соответствующего атрибута $Атрибут_i \subseteq$ Значение$_i$ $\subseteq$ *Входные параметры* $\subseteq Действие_k$, $Атрибут_i \subseteq$ Логические атрибуты объекта $\subseteq Объект_i$

В действии в описании наборов значений могут указываться только результаты (или оценки) и атрибуты, которые были определены в этом действии в "атрибутах объектов"

Ограничения событий

На одно состояние одного объекта не может быть установлено двух событий одинакового типа (при этом на одно состояние одного объекта могут быть установлены события разного типа).

Действие, указанное в событии, должно быть определено в множестве действий сцены.

Действие, указанное в событии, должно иметь тип "интерактивное".

Объект, указанный в событии, должен быть определен в множестве объектов сцены.

Декларативная модель интерактивных виртуальных сред с контролем

Декларативные модели данного типа состоят из множества объектов, множества действий и контролирующего сценария. На них распространяются все ограничения целостности, которые присутствуют у моделей виртуальных сред других типов.

Декларативная модель = <Объекты, Действия, Сценарий>

Если в сценарии в узле задан атрибут "входные параметры", то объект, входящий во множество объектов, заданных в этом атрибуте, должен входить во множество объектов среды.

Для любого объекта $Объект_i$, $\subseteq$ $Входной\ параметр_i$ $\subseteq$ *Входные параметры* $\subseteq$ $Вершина_k$ $Объект_i$ $\subseteq$ Объекты среды, 1 <= i <= количество входных параметров вершины, 1 <= k <= количество вершин сценария

Если в вершине задан атрибут "входные параметры", то атрибут объекта, указанный для объекта, входящего во множество входных параметров, заданных в этом атрибуте, должен входить во множество атрибутов этого объекта.

Для любого объекта $Объект_i$, $\subseteq$ $Входной\ параметр_i$ $\subseteq$ *Входные параметры* $\subseteq$ $Действие_k$ *и соответствующего атрибута* $Атрибут_i \subseteq$ Входные параметры$_i$ $\subseteq$ *Входные параметры* $\subseteq$ $Действие_k$, $Атрибут_i$ $\subseteq$ Атрибуты $\subseteq$ $Объект_i$

Для некоторых объектов, входящих во множество Объекты среды , указываются атрибуты, которые должны определяться в рамках обучающего задания. Для атрибутов можно указывать как конкретные значения, так и ссылки на существующие значения. Конкретное значение атрибута определит единственный вариант модели, связанный с этим атрибутом. Если необходимо сделать целое семейство подобных моделей с переменным атрибутом, ему необходимо в качестве значения указать интервал. Если атрибуту задан интервал, то в действиях, использующих этот атрибут, необходимо определить наборы вариантов значений для всего интервала.

На рис. 2.5.2 представлена схема отношений между атрибутами объектов, действиями и сценарием.

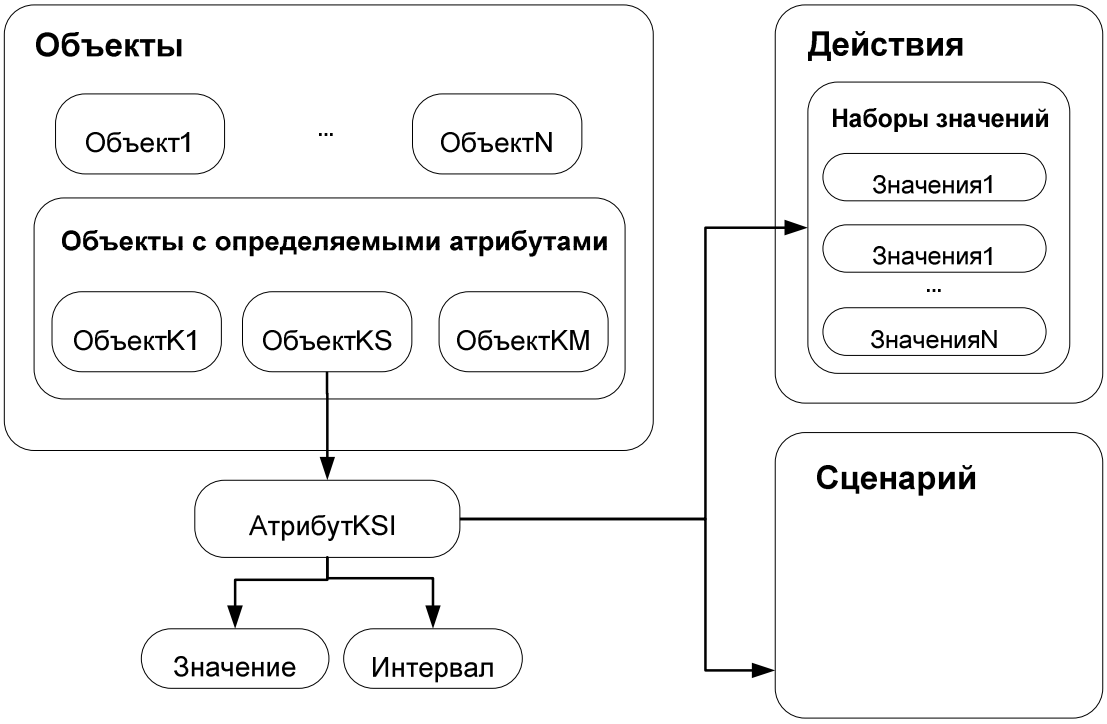


Рис. 2.5.2. Схема отношений между атрибутами объектов, действиями и сценарием.

Например, если для атрибута задан интервал дискретных значений <1-4>, то необходимо в действиях определять наборы данных для всех значений этого интервала, т.е. для значений: 1, 2, 3, 4. Для каждого перехода по результату в сценарии должен быть определен хотя бы один результат.

Для каждого перехода по результату в сценарии все результаты (или оценки результатов), указанные для этого перехода должны входить во множество результатов, указанных для действия, соответствующего узлу, из которого совершается переход.

Для каждого объекта, определенного в переходе по условию в сценарии, этот объект должен входить во множество {Объекты среды}.

Для каждого атрибута, определенного в переходе по условию в сценарии, этот атрибут должен входить во множество атрибутов объекта, который также указан в этом переходе по условию.

Для каждого объекта, определенного во множестве входных параметров этапа, этот объект должен входить во множество {Объекты среды}.

Для каждого атрибута, определенного во входных параметрах этапа, этот атрибут должен входить во множество атрибутов объекта, который также указан в этом входном параметре этапа.

2.6. Выводы

В данной главе в соответствии с первой задачей диссертационной работы сформированы основные принципы создания программного комплекса для разработки профессиональных виртуальных сред, разработана концептуальная архитектура программного комплекса. В соответствии с разработанными принципами проектирование, реализация и сопровождение профессиональной виртуальной среды (ПВС) на языке программирования заменяется проектированием и сопровождением декларативной модели ПВС с последующей ее интерпретацией; в процесс разработки включаются дизайнеры и эксперты предметной области, которые формируют модель ПВС по ее онтологии; средство разработки модели ПВС и сама виртуальная среда предоставляются разработчикам и пользователям как облачные сервисы.

В результате выполнения второй задачи диссертации, реализована онтология профессиональной виртуальной среды, состоящая из трех основных компонентов: объектов, действий и сценария, а также разработаны и классифицированы различные типы моделей профессиональных виртуальных сред. В онтологии выделено два уровня: логический и презентационный, позволяющие реализовать принцип включения в процесс разработки экспертов предметной области наряду с дизайнерами и разделению работы между ними. Декларативные модели представлены четырьмя основными типами: неинтерактивными статичными, неинтерактивными анимационными, интерактивными без контроля, интерактивными с контролем действий пользователя. Неинтерактивные статичные модели предназначены для демонстрации каких-либо предметов, статичных сцен. В неинтерактивных анимационных моделях объекты сцены имеют анимацию. Интерактивные виртуальные среды без контроля – среды, позволяющие пользователю взаимодействовать с объектами сцены, получая от них обратную реакцию.

Интерактивные с контролем – среды, в которых анализируются действия пользователя и выдается объяснение полученных результатов.

ГЛАВА 3. ИНТЕРПРЕТАТОР ПРОФЕССИОНАЛЬНОЙ ВИРТУАЛЬНОЙ СРЕДЫ

В настоящей главе решается задача разработки модели интерпретации виртуальной среды по ее декларативной модели. Рассматриваются клиент-серверная архитектура построения программного комплекса, возможные способы хранения и обмена данными, процесс перевода декларативной модели из терминов онтологии в программную форму представления.

3.1. Архитектура интерпретатора

Интерпретатор (см. рис. 3.1.1) состоит из функциональных блоков на клиенте, функциональных блоков на сервере и информационных ресурсов, используемых при интерпретации [38, 40].

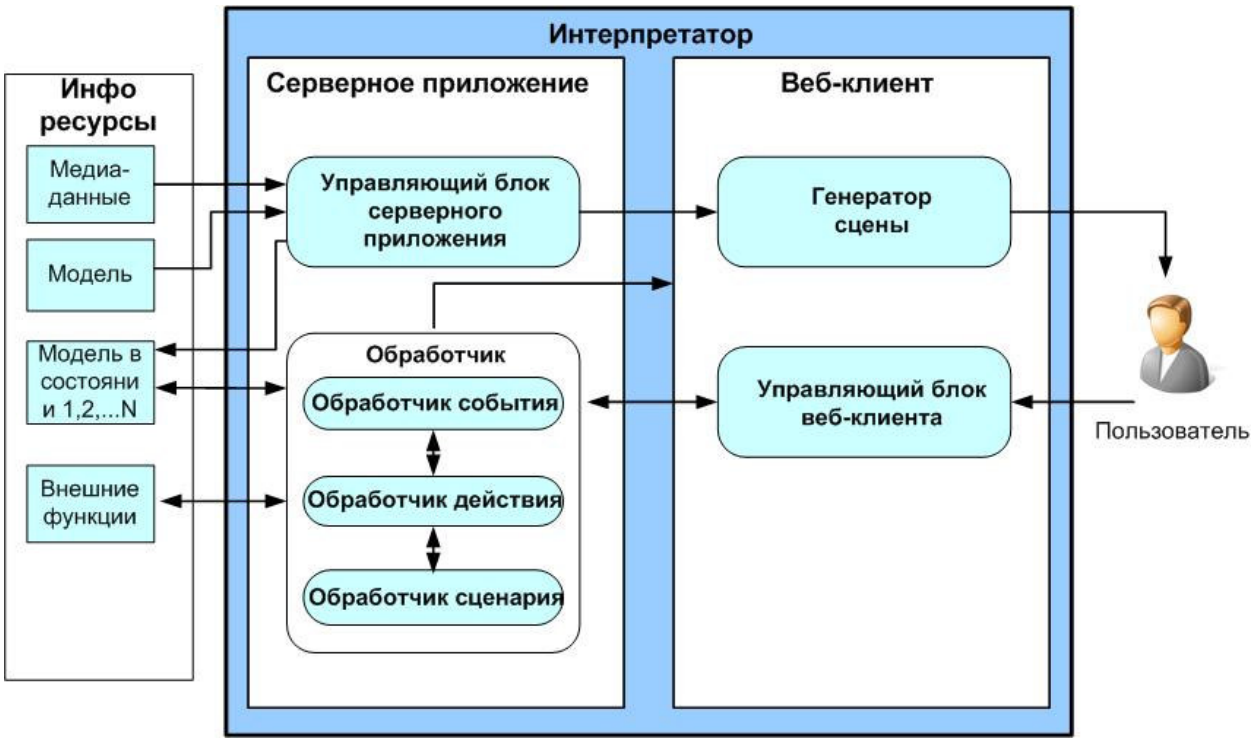


Рис. 3.1.1. Схема интерпретации декларативной модели

Функциональный блок веб-клиента выполняет отображение виртуальной сцены и взаимодействие с пользователем. Серверный функциональный блок выполняет логическую обработку данных.

Интерпретация начинается с загрузки декларативной модели и создания виртуальной сцены: создания объектов, привязки событий, установки начальных значений параметрам сцены.

Первым шагом является инициализация декларативной модели на сервере: запускается управляющий блок, который производит загрузку модели, ее преобразование, сохранение во временное состояние и передачу клиенту. На клиенте по переданной модели генератор сцены строит трехмерную виртуальную сцену: генерируются объекты сцены, загружаются необходимые медиа-данные (3d-модели, текстуры), на объекты добавляются анализаторы событий, устанавливается управление сценой. Сгенерированная сцена отображается пользователю, который может с ней взаимодействовать (взаимодействовать с объектами сцены), управляя подвижной камерой. Взаимодействуя с объектами сцены, а также вызывая необходимые команды напрямую, пользователь генерирует события. Эти события отправляются управляющему блоку клиента, который в зависимости от типа произведенного пользователем события осуществляет передачу события обработчику на сервер. Обработчик на сервере на основе текущей версии модели ПВС и полученной информации о событии и об объекте (если есть), связанным с этим событием, определяет необходимое для обработки действие. Обработчик действия на основе информации о действиях из текущей модели выполняет внутренние и, если необходимо, внешние функции обработки. Результатом этой обработки является изменение текущего состояния модели, запись нового состояния модели и отправка через управляющий блок веб-клиенту, который далее отображает изменения сцены пользователю. Кроме непосредственной обработки действия выполняется обработчик проверки действия на соответствие со сценарием, затем на клиент отправляется информационное сообщение с результатом. Каждое произведенное действие записывается в последовательность действий пользователя для дальнейшей обработки.

Рассмотрим далее более подробно процесс интерпретации каждого элемента представленной архитектуры с использованием декларативной модели.

Для последующего оформления отображения терминов онтологии в программное представление введем следующие обозначения:

=> - отображение онтологического термина в программное представление

{} - блок инструкций кода

3.2. Интерпретация на клиенте

Интерпретатор на клиенте выполняет следующие задачи:

- Построение виртуальной среды
- Обработка событий
- Изменение виртуальной среды
- Отображение информационных сообщений, объяснений

3.2.1. Построение виртуальной среды

Построение виртуальной среды состоит из трех шагов:

1. **Инициализация объектов**. Инициализация всех объектов сцены: создание 3d-моделей, наложение текстур.

2. **Инициализация действий**. Инициализация действий, установка связей между объектами и действиями.

3. **Инициализация связей между объектами, установка начальных состояний**. Инициализация ссылок между объектами, инициализация начальных состояний объектов.

Процесс инициализации представлен на рис. 3.2.1.

Инициализация объектов

На первом этапе создаются все объекты сцены, инициализируются все атрибуты уровня представления и атрибуты логического уровня.

Введем следующие функции интерпретатора виртуальной среды:

Создать_объект (имя, описание) - функция, создающая виртуальный абстрактный объект в виртуальной среде и добавляющая обработчики событий мыши.

Загрузить_модель (модель) - функция, загружающая трехмерную модель (трехмерный образ) объекта. Загруженная модель становится отображаемым образом объекта на виртуальной сцене.

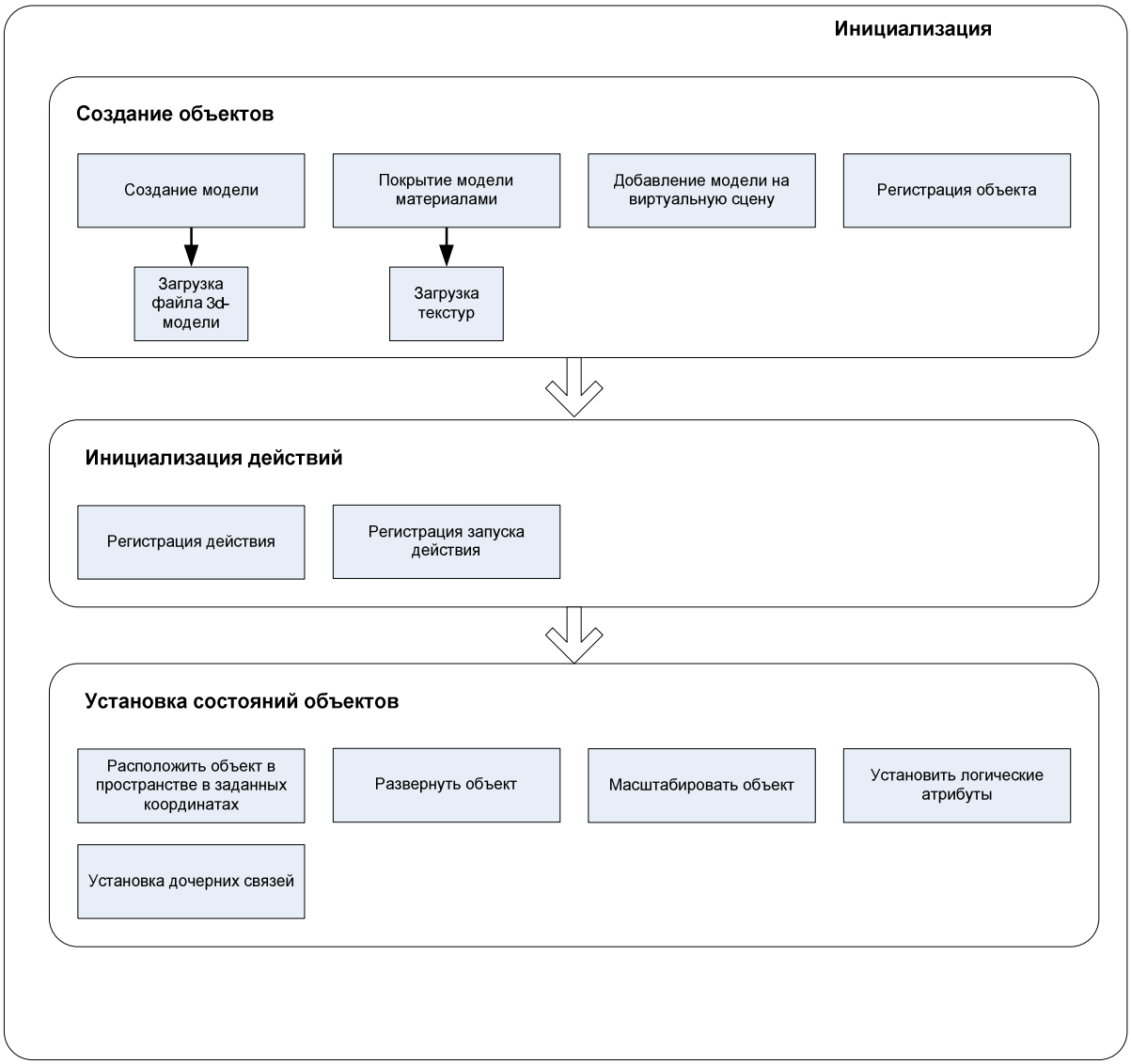


Рис. 3.2.1 Инициализация виртуальной среды

Загрузить_текстуры (текстуры) - функция, загружающая текстуры (покрытия) объекта. Загруженные текстуры накладываются на отображаемую модель объекта, чтобы придать ему более реалистичный вид.

Разместить_объект (x,y,z) - функция, размещающая объект в виртуальном пространстве сцены и помещающая его в координаты x,y,z.

Повернуть_объект (x,y,z) - функция, поворачивающая объект в виртуальном пространстве сцены по соответствующим осям x,y,z.

Масштабировать_объект (x,y,z) - функция, масштабирующая объект в виртуальном пространстве сцены по соответствующим осям x,y,z.

Создать_источник_света (цвет, сила, тень, тип) - функция, создающий источник света.

Создать_камеру () – функция для создания камеры, позволяющей пользователю перемещаться по виртуальной сцене.

Применяя приведенные выше функции и обозначения, выполняем отображение онтологии объектов в программное представление.

Простой объект = <Имя, Описание, Презентационные атрибуты> = <Имя, Описание, <Модель, Текстура, Координаты, Повороты, Масштабы>>

=>

{

Создать_объект (Имя, Описание)

Отобразить_простой_объект (Модель, Текстуры, Презентационные атрибуты)

}

Отображение объекта выполняется функцией:

Отобразить_объект (Модель, Текстуры, Презентационные атрибуты)

{

Загрузить_модель (Модель)

Загрузить_текстуры (Текстуры)

Трансформировать_объект (Презентационные атрибуты)

}

Трансформация объекта выполняется функцией:

Трансформировать_объект (Презентационные атрибуты)

{

Координаты = Презентационные атрибуты <Координаты>

Разместить_объект (Координаты<x>, Координаты<y>, Координаты<z>)

Повороты = Презентационные атрибуты <Повороты>

Повернуть_объект (Повороты<x>, Повороты <y>, Повороты <z>)

Масштабы = Презентационные атрибуты <Масштабы>

Масштабировать_объект (Масштабы <x>, Масштабы <y>, Масштабы <z>)

}

Изменяемый объект = <Имя, Описание, Логические атрибуты, Презентационные атрибуты, Множество состояний изменяемого объекта, Начальное состояние изменяемого объекта > = <Имя, Описание, <Модель, Текстура, Координаты, Повороты, Масштабы>, Множество состояний изменяемого объекта, Начальное состояние изменяемого объекта >

=>

{

Создать_ объект (Имя, Описание)

Отобразить_простой_объект (Модель, Текстуры, Презентационные атрибуты)

Для всех Состояние$_k$ ∈ {Множество состояний изменяемого объекта}$_i$ 1<= i <= количество состояний объекта: Зарегистрировать_состояние (Состояние$_k$)

Если (Начальное состояние != null) тогда

Отобразить_простой_объект (Модель, Текстуры, Презентационные атрибуты)

}

Составной объект = <Имя, Описание, Логические атрибуты, Презентационные атрибуты, Множество состояний составного объекта, Начальное состояние составного объекта > = <Имя, Описание, <Модель, Текстура, Координаты, Повороты, Масштабы>, Множество состояний составного объекта, Начальное состояние составного объекта >

=>

{

Создать_ объект (Имя, Описание)

Отобразить_простой_объект (Модель, Текстуры, Презентационные атрибуты)

Для всех Состояние$_k$ ∈ {Множество состояний изменяемого объекта}$_i$ 1<= i <= количество состояний объекта: зарегистрировать_состояние (Состояние$_k$)

Если (Начальное состояние != null) тогда

Отобразить_простой_объект (Модель, Текстуры, Презентационные атрибуты)

}

Для всех Объект$_k$ ∈ Дочерние объекты = {Простой объект, Изменяемый объект, Составной объект, Таблица, Источник света, Камера}$_i$ 0<= i <= количество дочерних объектов: установить_связь (Составной объект, Объект$_k$)

Таблица = <Имя, Описание, Логические атрибуты, Презентационные атрибуты, Множество состояний изменяемого объекта, Начальное состояние изменяемого объекта, Элементы > = <Имя, Описание, <Модель, Текстура, Координаты, Повороты, Масштабы>, Множество состояний изменяемого объекта, Начальное состояние изменяемого объекта >

=>

{

Создать_ объект (Имя, Описание)

Отобразить_простой_объект (Модель, Текстуры, Презентационные атрибуты)

Для всех Состояние$_k$ ∈ {Множество состояний изменяемого объекта}$_i$ 1<= i <= количество состояний объекта: зарегистрировать_состояние (Состояние$_k$)

Если (Начальное состояние != null) тогда

Отобразить_простой_объект (Модель, Текстуры, Презентационные атрибуты)

}

Источник света = <Имя, Описание, Логические атрибуты, Презентационные атрибуты источника света > = <Имя, Описание,

<Координаты, Повороты, Масштабы>, Множество состояний изменяемого объекта, Начальное состояние изменяемого объекта, Цвет, Сила, Тень, Тип >

=>

{

Создать_ объект (Имя, Описание)

Трансформировать_объект (Презентационные атрибуты)

Создать_источник_света (Цвет, Сила, Тень, Тип)

Для всех $Состояние_k \in \{Множество\ состояний\ изменяемого\ объекта\}_i$ *1<= i <= количество состояний объекта: зарегистрировать_состояние (*$Состояние_k$*)*

Если (Начальное состояние != null) тогда

Трансформировать_объект (Презентационные атрибуты)

}

Камера = <Имя, Описание, Логические атрибуты, Презентационные атрибуты источника света> = <Имя, Описание, <Координаты, Повороты, Масштабы>,>

=>

{

Создать_ объект (Имя, Описание)

Трансформировать_объект (Презентационные атрибуты источника света*)*

}

Объекты сцены в декларативной модели представлены в виде иерархического дерева, в котором каждая вершина – это некоторый объект одного из описанных выше типов: Простой, Изменяемый, Составной, Таблица, Источник света, Камера. Функция построения виртуальной сцены рекурсивно обходит это иерархическое дерево и строит по его вершинам объекты в соответствии с моделью интерпретации их метаинформации, представленной выше.

Таким образом, инициализация в интерпретации объектов будет выглядеть как отображение декларативной модели объектов (для каждого объекта) – в вызов соответствующего блока кода интерпретатора с вызовом абстрактных функций. На рис. 3.2.2 представлены элементы отображения различных типов объекта в абстрактные блоки кода.

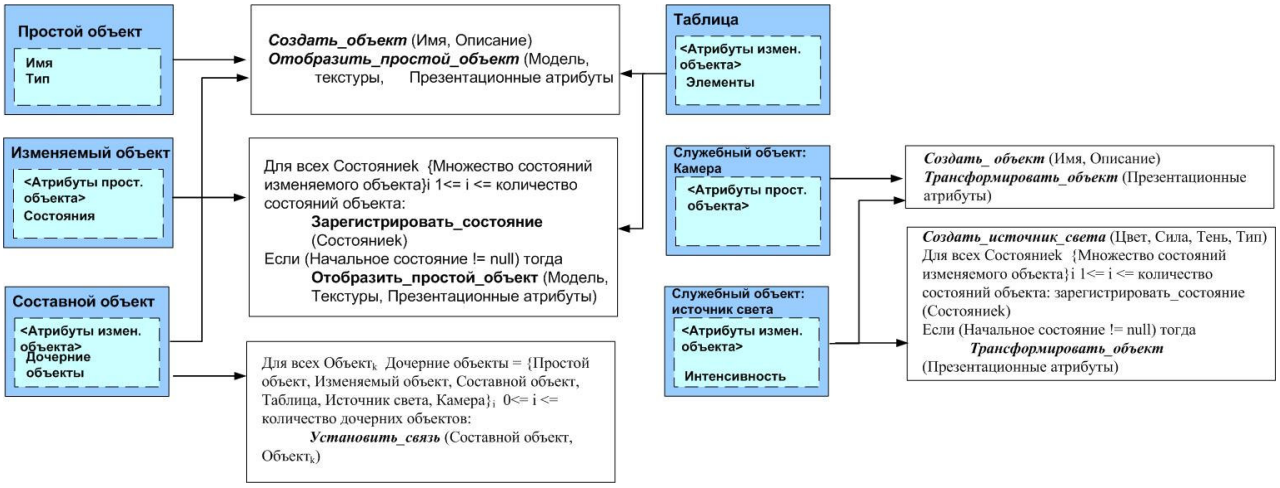


Рис. 3.2.2. Интерпретация объектов по декларативной модели

Инициализация действий

На втором этапе инициализируются действия сцены. Для каждого действия может быть указано событие, объект и состояние объекта, при котором на данное событие вызывается данное действие. Для объекта и его состояния добавляется событие и действие, которое должно вызываться.

Для каждого действия проверяется, задан ли у него запуск. Если запуск действия не указан, тогда инициализация действия заканчивается, иначе выполняется следующее:

объект -> найти этот объект среди объектов сцены

состояние -> найти это состояние среди состояний объекта

запуск -> установить для найденного объекта и его состояния этот тип запуска, при котором вызывать данное действие.

Инициализация связей между объектами и действиями

начальное состояние -> установить выбранное начальное состояние объекта из множества всех состояний.

Изменение состояния объекта (в том числе для начального состояния):

координаты -> расположить объект в трехмерном пространстве сцены с абсолютными координатами: координаты[0] -> x, координаты[1] -> y, координаты[2] -> z.

повороты -> развернуть объект в трехмерном пространстве сцены относительно трех перпендикулярных осей: повороты[0] -> x, повороты[1] -> y, повороты[2] -> z.

масштабы -> масштабировать объект в трехмерном пространстве сцены относительно трех перпендикулярных осей: масштабы[0] -> x, масштабы[1] -> y, масштабы[2] -> z.

дочерние объекты -> создать массив, каждый элемент которого будет ссылкой на другой объект сцены. При этом проверить наличие указанного объекта среди объектов сцены.

логические атрибуты -> установить значения указанных логических атрибутов.

При изменении состояния объекта выполняется поиск события в множестве событий сцены, для которого указан данный объект и изменяемое состояние. Если такое событие есть, то объекту добавляется обработчик этого события с вызовом действия, указанного в событии.

3.2.2. Обработка действий

В процессе работы с виртуальной средой пользователь находится в виртуальном пространстве сцены и выполняет некоторые операции (или действия) с виртуальными объектами. Задача интерпретатора – обрабатывать эти операции пользователя с виртуальными объектами: определять, какие именно действия должны запускаться в каждом конкретном случае.

В результате этих операций (или взаимодействия), объекты могут изменяться, переходить в новые состояния, отражая этим ответную реакцию на действия пользователя. Задача интерпретатора – определять, какие объекты должны измениться и как именно они должны измениться.

Кроме того, действия могут иметь информационные результаты (не связанные с непосредственным изменением объектов). Задача интерпретатора –

получить эти информационные результаты путем анализа текущих параметров выполняемого действия и отобразить их пользователю. Таким образом, можно выделить четыре основные задачи интерпретатора по обработке действий:

1. Выполнение (инициирование, идентификация, запуск) действия
2. Изменение объектов
3. Получение информационных результатов
4. Объяснение результатов

Рассмотрим каждую задачу подробнее.

Выполнение действия

Инициирование командных действий определяется однозначно выбором пользователя через интерфейс интерпретатора. Интерактивные действия запускаются путем взаимодействия пользователя с объектами виртуального мира.

С объектами может быть связано множество действий или вообще ни одного действия. В один момент времени с объектом может быть связано не более одного действия. Если с объектом связано более одного действия, то какое именно действие будет выполняться, устанавливается и определяется в состоянии, в котором объект находится в настоящий момент. Если состояние объекта меняется, то может измениться его связь с действием.

Например, пусть есть изменяемый объект с двумя состояниями: "включен", "выключен" и есть два соответствующих действия "включить" и "выключить". Тогда, если объект находится в состоянии "включен", к нему можно применять действие "выключить", а когда он выключен, тогда наоборот – "включить". Таким образом, в процессе работы программы устанавливаются динамические связи объектов и действий.

На этапе инициализации с каждым действием связываются событие, объект и состояние объекта.

Изменение объектов

Для того чтобы изменять объекты, в структуре действия есть информация о том, какие объекты должны принять требуемые состояния.

состояния объектов -> пройти по заданному массиву и для каждого элемента изменить состояние объекта:

<объект, состояние> -> изменить состояние объекта **объект** на **состояние**. Изменение состояния происходит также как установка начального состояния: если необходимо, у объекта меняются координаты, повороты, логические атрибуты и т.д.

Получение информационных результатов

Информационные результаты действий могут зависеть от множества различных параметров, включающих атрибуты объектов, параметры сцены в целом. Алгоритм получения результата может быть сложным и нестандартным, реализованным во внешнем модуле. Поэтому основная тяжесть задачи получения информационных результатов переносится на серверную часть, на которую с клиента отправляется соответствующее сообщение.

Таким образом, задача обработки информационных результатов на клиенте состоит из следующих подзадач:

1. Сформировать сообщение серверу о необходимости выполнения некоторого алгоритма для получения результата
2. Получить ответное сообщение от сервера с результатом выполнения.
3. Отобразить пользователю результат обработки

Задача интерпретатора – найти результат, соответствующий всем полученным параметрам, отобразить его пользователю в виртуальной среде, получить оценку этого результата и передать на обработку в сценарий.

Объяснение результатов

В некоторых случаях может потребоваться генерация объяснения получившихся результатов обработки пользователю на основе текущих данных и данных сценария.

На рис. 3.2.3 представлена схема обработки действий, возвращающих пользователю результат.

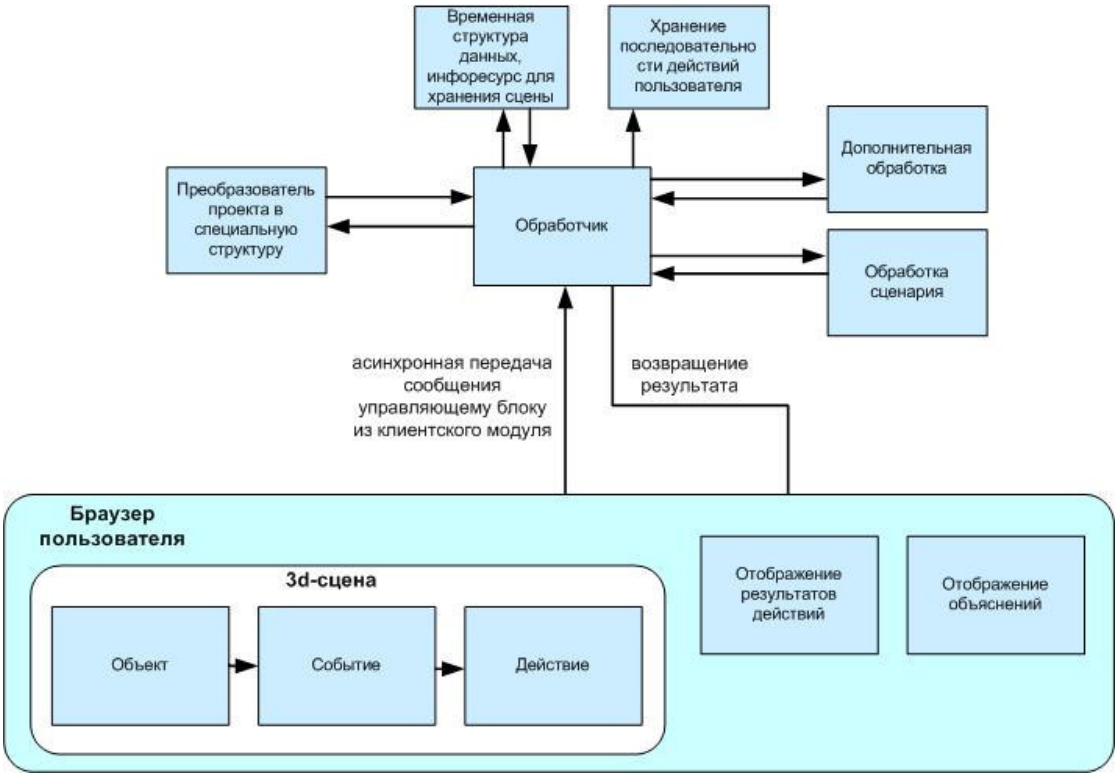


Рис. 3.2.3. Обработка действий

3.3. Интерпретация на сервере

3.3.1. Загрузка проекта и отправка его клиенту

Первая задача, с которой сталкивается пользователь, а, следовательно, и система, - это получение необходимой пользователю виртуальной среды.

Необходимо получить требуемую модель, сохраненную на сервере; и передать клиенту.

3.3.2. Обработка действий, переданных с клиента

Находясь в пространстве виртуальной интерактивной среды на клиенте, пользователь производит некоторые действия (события), для решения которых может потребоваться нестандартная внешняя обработка, описанная разработчиками среды в отдельных модулях на сервере.

Задача серверного обрабатывающего модуля: получить переданное с клиента сообщение, из которого извлечь информацию о произведенном действии пользователя или об объекте и событии, извлечь дополнительные параметры действия. Выполнить стандартную обработку действия, обновить

проект сцены в локальном хранилище на сервере для синхронизации с проектом сцены на клиенте, вызвать заданный внешний модуль обработки.

Стандартная обработка действия

Стандартная обработка действия, выполняемая на сервере, происходит на основе данных, указанных в модели для данного действия.

Информационные результаты действий могут зависеть от множества различных параметров, включающих атрибуты объектов, параметры сцены в целом. Задача интерпретатора – найти результат, соответствующий всем полученным параметрам, отобразить его пользователю в виртуальной среде, получить оценку этого результата и передать на обработку в сценарий.

Для получения результата выполняется следующее:

параметры результата -> получение текущих значений всех указанных атрибутов.

наборы значений -> выполнение обработки для каждого набора из множества.

набор значений -> сопоставление всех атрибутов набора с полученными текущими параметрами результата. Набор, значения атрибутов которого соответствуют текущим параметрам, считается результирующим, и результат (или оценка результата) этого набора считается результатом обработки.

Если в результирующем наборе задан результат, вызывается функция отображения полученного результата для пользователя.

Если в проекте задан сценарий, то результат передается на обработку в сценарий.

Если в результирующем наборе вместо результата задана оценка результата, то для получения непосредственного результата вызывается обрабатывающая функция, заданная для действия.

результат -> получение результата на основе оценки с помощью функции.

Если в проекте задан сценарий, то на обработку в сценарий передается не результат, а оценка результата.

3.3.3. Вызов внешних функций для обработки событий

Внешние функции должны, с одной стороны, работать максимально независимо от остальной части программы (основного модуля и других внешних модулей) и, с другой стороны, должны при необходимости иметь доступ к полному описанию текущей модели. Поэтому внешние функции должны реализовываться в рамках общей серверной платформы.

Для обеспечения независимости разных внешних модулей и основного модуля друг от друга их совместная работа должна осуществляться путем передачи сообщений. Каким образом устроен каждый отдельный внешний модуль – должно быть "черным ящиком" для других модулей.

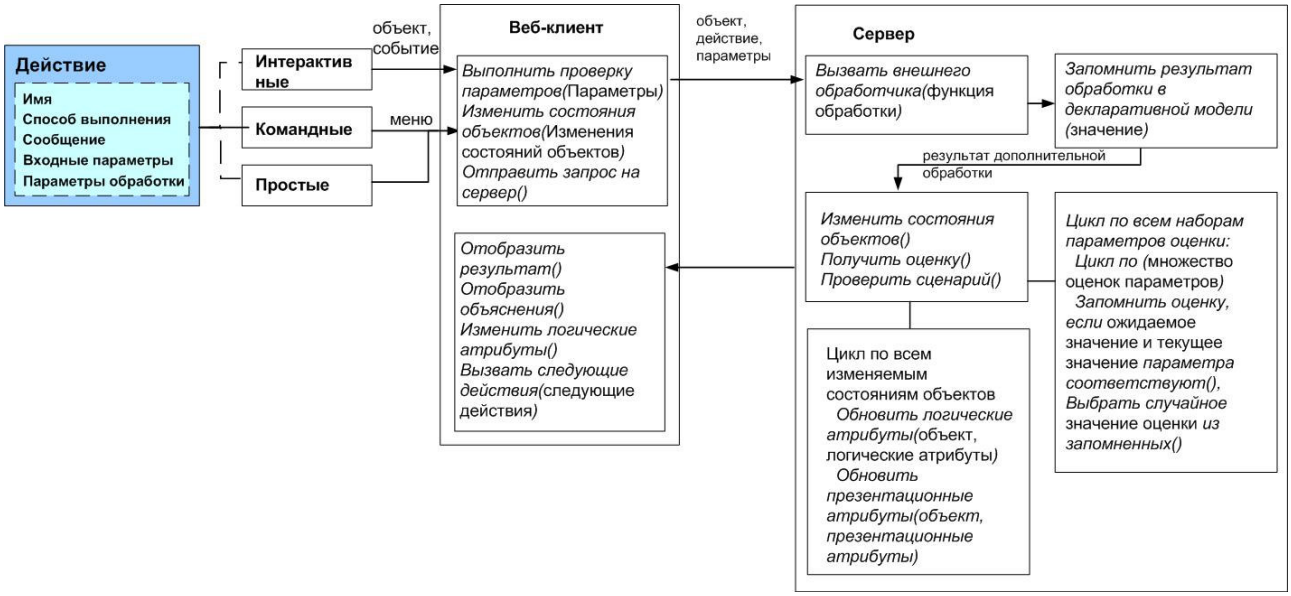


Рис. 3.3.1. Интерпретация действий

На рис. 3.3.1 представлена обработка действий в виде абстрактных функций, работающих на основе декларативной модели на веб-клиенте и сервере, схема передачи запросов и данных. Продемонстрировано, каким образом используются атрибуты и компоненты декларативной модели в вызове функций обработки действий.

Внешние модули должны иметь доступ ко всем параметрам события, с которого они вызываются (например, координаты мышки). Для этого все параметры сообщения, полученного с клиента в формате множества,

передаются в полном объеме другим внешним модулям, которые уже сами разбираются, какие из параметров имеют необходимость.

3.3.4. Обработка сценария для выполненного действия

В модели может быть задан сценарий, в соответствии с которым осуществляется проверка действий пользователя.

Обработка сценария включает: хранение данных о текущем действии; по информации от переходов определение следующего действия; определение узла сценария в случае неправильного действия пользователя.

Обработка сценария начинается непосредственно после завершения обработки действия пользователя. Обработчику сценария передается информация о выполненном действии и о результатах, полученных в действии.

Сначала определяется, выполнил ли пользователь хотя бы одно из ожидаемых от него действий. Для этого сравнивается информация о выполненном действии и обо всех ожидаемых действиях.

Информация о выполненном действии поступает в обработчик сценария после выполнения действия. Информация о каждом ожидаемом действии включает:

имя действия – имя ожидаемого действия. Имя ожидаемого действия сравнивается с именем выполненного действия, если они совпадают, тогда выполняется проверка параметров действия. Если нет, тогда выполняется поиск другого возможного следующего узла.

параметры действия – значения атрибутов объектов сцены, которые ожидаются по сценарию в момент совершения действия. Для выполнения проверки обрабатываются все параметры, указанные в узле; для каждого параметра находится значение атрибута объекта, указанного в параметре и сравнивается с указанным значением для параметра. Если все параметры совпадают, тогда поиск заканчивается успешно и считается, что пользователь выполнил верное действие. Если нет, тогда выполняется поиск другого возможного следующего узла.

Если и имена, и параметры ожидаемого и выполненного действия совпали, тогда выполняется переход к данному узлу, т.е. данный узел становится текущим, и осуществляется поиск следующих возможных ожидаемых узлов на основании информации о переходах для нового узла и полученных результатов действия.

Если имена и параметры ни одного из ожидаемых действий не совпали с именем и параметрами выполненного действия, то пользователю сообщается о неверном действии и осуществляется поиск узла, с которого можно продолжить проверку следующих действий пользователя.

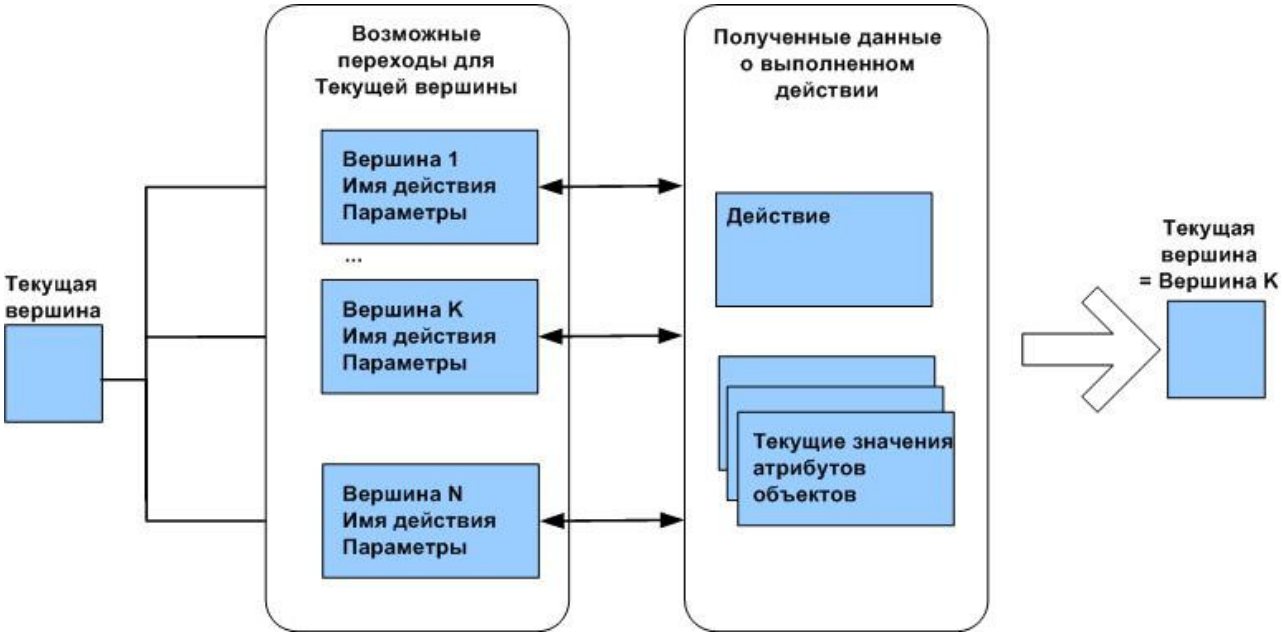


Рис 3.3.2. Нахождение верного перехода к новой Текущей вершине

На рис. 3.3.2 показано, что у Текущей вершины есть множество "Возможные переходы для Текущей вершины". Каждый переход из этого множества проверяется на соответствие с "Полученными данными о выполненном действии" и, таким образом находится переход K, ведущий к ВершинеK, которая становится новой Текущей вершиной, для которой уже в свою очередь формируется множество "Возможные переходы для Текущей вершины".

Для поиска следующего возможного ожидаемого узла проверяется набор переходов текущего (выполненного) узла.

набор переходов -> выполняется обработка каждого перехода в наборе.

Если тип перехода – фиксированный, то он помещается в набор возможных следующих переходов.

Если тип перехода – по результату, тогда полученный в сценарии результат прошлого действия сравнивается с результатом для данного перехода. Если полученный результат соответствует результату перехода, тогда данный переход помещается в набор возможных следующих переходов. Если полученный результат действия не соответствует результату перехода, тогда этот переход пропускается.

Если тип перехода – по условию, тогда осуществляется проверка условия, и если оно выполняется, тогда данный переход помещается в набор возможных следующих переходов.

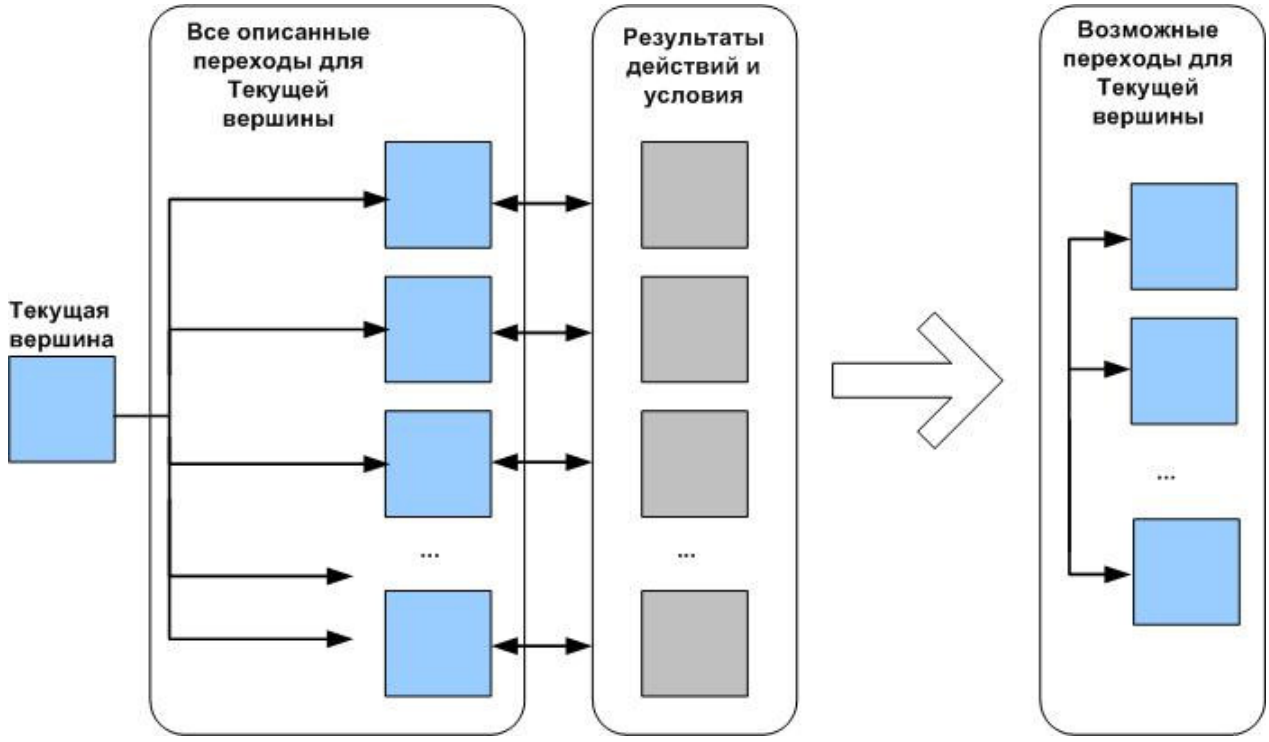


Рис. 3.3.3. Получение возможных следующих переходов для Текущей вершины

На рис. 3.3.3 показан процесс построения множества "Возможные переходы для Текущей вершины". Среди множества "Все описанные переходы для Текущей вершины" выбираются те, которые соответствуют полученным результатам действия и/или дополнительным условиям, заданным при

переходах; фиксированные переходы заносятся в множество возможных переходов без каких-либо проверок.

3.3.5. Обновление информации о текущем состоянии модели

Все изменения в модели происходят в результате выполнения действий. Информации о действии и о текущем состоянии модели достаточно для правильного обновления, которое, таким образом, может и должно производиться асинхронно отдельно на клиенте и отдельно на сервере.

3.4. Хранение и обмен информацией

Декларативная модель включает в себя информацию о структуре виртуальной интерактивной среды на сервере и на клиенте, информацию об изменениях сцены, информацию для обмена между клиентом и сервером.

На клиенте информация о структуре сцены необходима в любой момент времени, так как вся сцена строится и изменяется на основе этой информации, обработка событий также зависит от текущей структуры сцены.

На сервере информация о структуре сцены необходима для внешней обработки действий, выполняемых через сервер и внешние модули, так как любая обработка действий зависит от текущего состояния объектов сцены.

Виртуальная интерактивная среда динамически изменяется, начиная от некоторого стартового состояния, заданного в загружаемой модели. Для динамического изменения виртуальной среды необходимо хранить и изменять текущую структуру всей сцены. Структура всей сцены может содержать множество различных данных, при этом уникальных для каждого конкретного пользователя и для каждой запущенной им ПВС. Таким образом, появляются следующие дополнительные задачи по обработке информации, содержащейся в декларативной модели:

1) хранение структуры сцены

2) обмен информацией между клиентом и сервером

Данные задачи являются взаимосвязанными, так как, увеличив сложность одной, можно уменьшить сложность другой.

Обмен информацией о полной структуре сцены избыточен, требует большого объема пересылаемых данных. Для синхронизации информации о структуре сцены между клиентом и сервером достаточно пересылать информацию об изменениях или о выполняемых действиях.

Для обмена информацией между двумя раздельными частями возможно два подхода: синхронный и асинхронный. Синхронный подход означает прекращение любых операций до тех пор, пока информация не станет одинаковой на обоих концах. Этот подход гарантирует стабильность работы программы, так как есть полная уверенность, что данные везде одинаковы и соответствуют друг другу. Однако, в данном подходе возникает необходимость ожидания синхронизации после каждого действия, что сильно замедляет процесс работы с программой. Другой подход – асинхронный. Асинхронный обмен информацией означает отсутствие ожидания подтверждения о доставке и полном соответствии информации. В асинхронном подходе информация отправляется и работа продолжается, ничего не останавливается для ожидания. Этот метод лучше подходит для виртуальных интерактивных сред, так как позволяет достичь значительно более высоких результатов интерактивного взаимодействия в режиме реального времени. Таким образом, обмен информацией между клиентом и сервером должен происходить в асинхронном режиме.

3.5. Выводы

В результате выполнения третьей задачи диссертационной работы были разработаны методы интерпретации декларативной модели в соответствии с принципом автоматизации разработки профессиональных виртуальных сред. Методы интерпретации модели профессиональных сред включают в себя создание виртуальной сцены, обработку событий, изменение объектов сцены. Определяется механизм взаимодействия между клиентской и серверной частью виртуальной среды, форматы сообщений между различными модулями, механизм хранения данных.

ГЛАВА 4. МЕТОДЫ РЕАЛИЗАЦИИ ПРОГРАММНОГО КОМПЛЕКСА

В данной главе формулируются требования к программному комплексу, обосновывается выбор технологии и средства его реализации, а также описывается методы реализации каждого компонента программного комплекса в соответствии с выбранной технологией.

4.1. Требования к реализации программного комплекса

Во второй главе была описана концептуальная архитектура программного комплекса для разработки и использования профессиональных виртуальных сред (ПВС). В данном разделе формулируются требования к реализации программного комплекса, основанные на анализе обзора литературы, инструментальных средств для создания виртуальных сред общего назначения, а также прикладных программных систем.

Для разработчиков:

Программный комплекс должен:

1. обеспечить создание модели ПВС с помощью редакторов , реализованных как облачные сервисы без необходимости скачиваний и установок программного обеспечения комплекса;
2. предоставить средства авторизации и ограниченного доступа к инструментам создания и редактирования модели ПВС;
3. предоставить специализированные редакторы различным группам разработчиков для создания модели ПВС:
    a. экспертам – структурный редактор, управляемый онтологией, для создания логического представления модели ПВС;
    b. дизайнерам – графический 3-D редактор для создания презентационного представления модели ПВС;
4. обеспечить проверку целостности модели ПВС;
5. удаленно (на внешнем ресурсе) хранить и загружать информационные и программные компоненты ПВС модель, файлы медиа-ресурсов (3d-моделей, текстур и т.п.), программные модули;

6. обеспечить возможность повторного использования программных и информационных компонентов.

Требование 1 следует из 3-го принципа разработки программного комплекса (*Использование как средства разработки ПВС, так и готового приложения как интернет-сервисов*). Для работы с программным комплексом специалистам необходим компьютер и выход в Интернет с доступом к сервисам инструментального средства без необходимости настройки какого-либо оборудования, скачивания "тяжелых клиентов" и т.д.

Требование 2 основывается на том, что к удаленной разработке современного программного обеспечения группой специалистов не должны иметь доступа посторонние лица.

Требование 3 следует из 2-го принципа разработки программного комплекса (*Включение в процесс разработки специалистов разного профиля, экспертов предметной области, дизайнеров, программистов и разделение работы между ними.)*

Требования 4, 6 следуют из 1-го принципа разработки программного комплекса, (*Автоматизация процесса разработки профессиональной виртуальной среды*) и предполагают упрощение разработки за счет повторной используемости и автоматической проверки целостности.

Требование 5 является стандартным требованием к веб-приложениям, т.к. во многих из них активно используются различные медиа-файлы и ресурсы, которые в соответствии с данным требованием, должны находиться на сервере сервиса, а не на локальных компьютерах разработчиков или пользователей, и к которым должен обеспечиваться быстрый доступ. Разработчики сервиса должны иметь возможность сохранять и использовать загружаемые на сервер файлы.

Для пользователей:

1. обеспечить работу всех средств редактирования и разрабатываемых ПВС как облачных сервисов, без необходимости скачиваний и установок программного обеспечения комплекса

2. предоставить средства авторизации и регулируемого доступа к профессиональным виртуальным средам

3. предоставить средства интерфейса для ПВС: выбор, запуск, инициализация, интерактивное взаимодействие с виртуальной средой, выход

4. обеспечить работу ПВС в режиме сессии, т.е. программное приложение должно работать непрерывно, интерактивно реагируя на действия пользователя, до тех пор, пока он не закроет приложение

5. иметь средства отображения и объяснения результатов действий пользователя.

Требование 1 для пользователей является аналогичным требованию для разработчиков**.**

Требование 2 является стандартным требованием к удаленному личному использованию современного программного обеспечения, к использованию которого не должны иметь доступа посторонние лица.

Требования 3, 4, 5, являются требованием к принципиальным функциональным интерфейсным возможностям сервиса, позволяющим пользователям выбирать настройки виртуальных сред и получать информацию об их работе.

а. Технологии для реализации программного комплекса

В данном разделе рассматриваются возможные технологии для реализации программного комплекса, обосновывается выбор конкретной технологии для его реализации.

4.2.1. Технологии реализации серверной части программного комплекс

В качестве возможных средств разработки серверной части программного комплекса рассматривались следующие варианты: 1) традиционный подход (веб-сервер + язык программирования + СУБД) [44, 57, 55, 58, 101], 2) облачная платформа IACPaaS [26-28].

Первый подход – это стандартный подход к разработке веб-приложений в Интернете. В качестве языков программирования могут быть: PHP, Perl, C# и т.д., в качестве СУБД - MySql, PostgreSQL, MSSQL и т.д.. В данном подходе

веб-сервисы создаются как набор динамически генерируемых веб-страниц, написанных на выбранном языке программирования, с обращением к информационным данным с помощью выбранной СУБД [44, 57, 55, 58, 101].

Второй подход предполагает использование облачной платформы IACPaaS, разработанной в лаборатории интеллектуальных систем Института автоматики и процессов управления ДВО РАН. Платформа IACPaaS предлагает совершенно другую инфраструктуру для создания и сопровождения веб-сервисов на основе мультиагентного подхода и семантических сетей [25].

Рассмотрим, каким образом, каждый из подходов может быть использован в реализации некоторых из поставленных выше требований к программному комплексу.

Традиционный подход (веб-сервер + язык программирования + СУБД)

1) Создание веб-сервисов внутри традиционного сайта требует значительных усилий даже для простых сервисов, т.к. для них нужна соответствующая сервисам инфраструктура и программные модули. Для реализации интерактивных сервисов возникает необходимость программировать работу с сессиями.

2) Существуют отдельные библиотеки, упрощающие авторизацию пользователей, однако в случае их использования также требуется программная настройка, т.е. в программных модулях необходимо прописывать соответствующие инструкции доступа и использования библиотек.

3) Существуют редакторы, ориентированные на реляционные СУБД, существуют редакторы для работы с базами знаний (Protege), однако, связывать такие реляционные данные и знания из таких баз с программными модулями достаточно сложно, так как это выполняется программным путем.

4) Имеется возможность в хранении файлов и ресурсов на сервере. Однако, разработчикам сервисов требуется самим производить мониторинг сохранности ресурсов и ограниченного доступа к ним.

5) Организация одновременной работы с несколькими запущенными сервисами или программными единицами, работающими с одними и теми же наборами данных, требует трудоемкого программирования, учитывающего множество сложных конфликтных ситуаций одновременного использования данных.

6) Есть возможность создать непрерывно работающее веб-приложение, однако для этого требуется реализовать механизм сессий.

Облачная платформа IACPaaS

1. Облачность [56] платформы IACPaaS определяет структуру платформы для создания веб-сервисов. При этом механизм создания сервисов заключается в работе с интерфейсом платформы.

2. На платформе реализована сложная инфраструктура, включающая гибкие средства авторизации пользователей.

3. Имеются визуальные средства создания и редактирования данных и знаний, на основе онтологий. В частности имеется один из требуемых для программного комплекса редактор – редактор информационных ресурсов.

4. Платформа IACPaaS предлагает развитую инфраструктуру для хранения, использования, управления данными и знаниями, которые хранятся в форме семантических сетей и представлены инфоресурсами.

5. Платформа IACPaaS использует мультиагентный подход, что обеспечивает упрощение управления и повторного использования программных модулей, сервисов и знаний системы.

6. Предоставляет возможность одновременной работы с несколькими запущенными приложениями, при этом платформа берет на себя ответственность за решение конфликтных ситуаций по одновременному использованию ресурсов.

7. Есть возможность создать непрерывно работающее веб-приложение, при этом не требуется реализовывать механизм сессий. По всем приведенным критериям рассмотренных средств видно, что платформа IACPaaS предоставляет лучшие возможности для реализации поставленных требований.

Таким образом, в результате анализа обоих рассмотренных подходов и с учетом требований к разработке системы для ее реализации было решено использовать именно облачную платформу IACPaaS.

4.2.2. Технологии реализация клиентской части программного комплекса

Разработка профессиональных виртуальных сред предполагает непосредственную работу с трехмерной графикой. Платформа IACPaaS является общим окружением и инфраструктурой для программных сервисов и является средством реализации серверной части, поэтому в ней не заложены такие специфические функциональные возможности, как программирование и отображение трехмерной графики. Однако платформа позволяет встраивать в сервисы сторонние программные модули, позволяющие реализовать практически любые требования, нереализованные в самой платформе. Таким образом, для отображения профессиональных виртуальных сред в графическом виде необходим специальный внешний программный модуль. Важным требованием к такому внешнему программному модулю является возможность работать через Интернет (как веб-клиент), чтобы иметь возможность быть встроенным в облачный сервис на платформе.

Для разработки веб-клиента рассматривались различные технологии: DirectX, OpenGL, Unity3D, Flash, Silverlight [118] и другие.

Технологии на DirectX, OpenGL, Unity3D и другие

Технологии (движки) на DirectX [105], OpenGL [5, 85] имеют хорошую производительность и качество графики, хорошую поддержку и документацию. Однако они предполагают разработку «стандартных» десктопных клиентов на C++ [72], C#, Java (т.е. не могут быть "веб-клиентом") и требуют достаточно сложного программирования на этих языках.

Unity3D [109] также имеет хорошую производительность и качество графики и, кроме того, изначально рассчитан на разработку Интернет-приложений (т.е. может быть "веб-клиентом"). Однако, плагин Unity недостаточно хорошо распространен среди пользователей Интернета.

Flash-технология

Flash - это мультимедийная платформа компании Adobe для создания веб-приложений. Flash Player представляет собой виртуальную машину, на которой выполняется загруженный из Интернета код flash-программы. Flash-файл (в формате swf) хранится в фонде платформы, передается и отображается в браузере, встраиваясь, как и любое другое медиа-содержимое (картинки, видео-файлы) в открытую страницу браузера.

Технология Flash позволяет генерировать, отображать и интерактивно взаимодействовать со сложными трехмерными графическими сценами на приемлемой скорости непосредственно в браузере, благодаря прямому использованию видеокарты компьютера пользователя.

Преимущества технологии Flash [112, 114]:

- Плагин Flash установлен у большинства пользователей Интернета (более 90 %). Это означает доступность и хорошую распространяемость Flash-приложений.
- Совмещение технологий растровой и векторной графики, что позволяет существенно сокращать объемы памяти, расходуемой графическими приложениями на Flash.
- Наличие мощной базовой платформы [114], языка программирования и богатого функционала для разработки Интернет-приложений на Flash.
- Полноценная поддержка аппаратно-ускоренной трехмерной компьютерной графики.
- Присутствие большого количества различных бесплатных средств разработки (включая библиотеки, редакторы, фреймворки).[Parr A., AS3 Code Libraries]

Таким образом, на основе проведенного анализа был сделан выбор в пользу технологии Flash для реализации веб-клиента системы. Реализация веб-клиента графического выполнена на языке ActionScript 3.0 [60, 63, 137]. Веб-клиент представляет из себя Flash-файл (в формате swf), который хранится в фонде платформы и передается пользователю в браузер, встраиваясь, как и любое другое медиа-содержимое (картинки, видео-файлы), в открытую

страницу браузера. Flash веб-клиент загружается и встраивается в веб-страницу сервисного приложения сразу после запуска выбранного сервиса приложения виртуальной среды незаметно для пользователя аналогично подгрузке картинок в html-страницы. В это же время веб-клиенту передаются от сервисного приложения все необходимые данные: декларативная модель виртуальной среды, соответствующая запущенному сервису; медиа-данные, включая трехмерные модели, текстуры, звуки и т.д.

Для реализации трехмерной графики на Flash была выбрана библиотека Alternativa3D. В [122] описываются основные преимущества и возможности данной библиотеки. С помощью данной библиотеки на Flash в интернете появляется возможность создавать современный уровень трехмерной графики, включая миллионы используемых полигонов, свет, тени и шейдеры [78, 79].

4.3. Платформа IACPaaS

Платформа для разработки интернет-сервисов представляет собой программно-информационный интернет-комплекс IACPaaS (Intelligence Application, Control and Platform as a Service), предоставляющий контролируемый доступ и единую систему администрирования для создания и использования интеллектуальных сервисов и их компонентов, представленных семантическими сетями, поддержку функционирования агентов (через передачу и обработку сообщений, запуск методов).

Интернет-комплекс IACPaaS основан на технологии облачных вычислений и обеспечивает удаленный доступ конечным пользователям к интеллектуальным системам, а разработчикам и управляющим – к средствам создания интеллектуальных систем и управления ими.

Основными архитектурными компонентами интернет-комплекса IACPaaS являются: фонд, административная система и виртуальная машина (см. рис. 4.3.1).

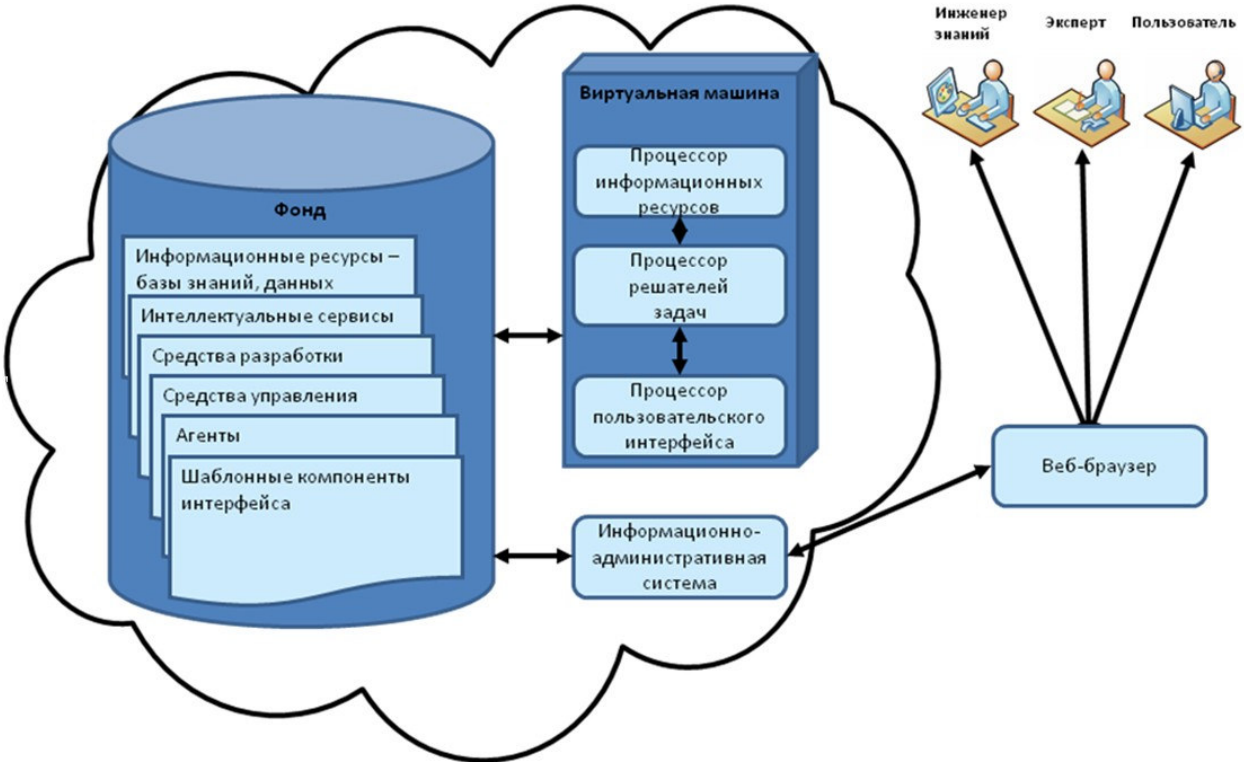


Рис. 4.3.1. Концептуальная архитектура платформы IACPaaS

Фонд представляет собой совокупность единиц хранения – программных и информационных ресурсов различных типов; для удобства навигации он разделен на предметные области, а те, в свою очередь, на разделы; каждый раздел содержит относящиеся к нему единицы хранения: прикладные и инструментальные средства (средства разработки и управления), их агенты, информационные ресурсы.

Административная система предназначена для всех пользователей проекта. С ее помощью они могут просматривать доступное им содержимое фонда; подавать заявки на регистрацию в предметных областях фонда, на регистрацию полномочий на использование системных и прикладных интернет-сервисов платформы, на модификацию фонда, а также реализовывать свои полномочия.

Виртуальная машина представляет собой совокупность трех процессоров (процессора информационных ресурсов, процессора решателей задач, процессора пользовательского интерфейса), а также ряда вспомогательных средств. Каждый процессор представляет собой набор функций для поддержки соответствующих компонентов интернет-сервиса.

Процессор информационных ресурсов представляет собой набор функций обработки информационных ресурсов фонда, доступный разработчикам и сопровождающим виртуальной машины, а также разработчикам и сопровождающим кода программных компонентов. Все информационные ресурсы имеют единое унифицированное декларативное представление – в виде семантической сети.

Процессор решателей задач состоит из инициализатора и коммуникационной системы. Инициализатор осуществляет запуск прикладных и системных сервисов. Коммуникационная система представляет собой набор функций, предназначенный для активизации агентов решателей задач, а также формирования и передачи сообщений между ними.

Процессор пользовательского интерфейса состоит из совокупности интерфейсных агентов и интерпретатора интерфейса. Интерфейсные агенты формируют описание абстрактного интерфейса пользователя в соответствии с моделью абстрактного интерфейса. Интерпретатор интерфейса выполняет построение конкретного пользовательского интерфейса на основе описания абстрактного интерфейса и модели конкретного интерфейса.

Все сервисы платформы реализуются на основе мультиагентного подхода. Мультиагентный интернет-сервис представлен своим декларативным описанием, которое включает в себя, в общем случае: (1) входные, выходные и собственные информационные ресурсы различного уровня общности (содержательно это могут быть онтологии, базы знаний, базы данных и т.д.), (2) решатель задач, представленный описанием множества агентов и, возможно, управляющим графом, и (3) пользовательский интерфейс. Агент – повторно используемый программный компонент, взаимодействующий с другими агентами посредством приема и передачи сообщений. Агент состоит из двух частей – декларативной и процедурной. Декларативная часть представляет собой спецификацию агента – информационный ресурс, в котором описана структура агента и которое должно использоваться средствами облачной платформы для автоматической генерации шаблона его исходного кода, а также

для его внедрения в общую инфраструктуру платформы. Процедурная часть (код агента) – есть код консеквентов всех его продукций.

Агент включает блоки продукции (методы), выполняющие обработку принимаемых им сообщений с целью решения связанной с агентом подзадачи. Для каждого блока продукций определен свой шаблон сообщения. Результатом работы метода может быть формирование и посылка сообщения конкретному агенту, либо множеству агентов, а также формирование или модификация некоторого информационного ресурса.

Продукция состоит из условия (антецедента) и действия (консеквента). Антецедент такой продукции используется для анализа сообщения, выбора применимого метода и передачи ему информации из сообщения; консеквент, по сути, является вызовом метода.

Все онтологии (шаблоны) сообщений содержатся в фонде IACPaaS. Для передачи сообщения некоторому агенту приложения используется готовый шаблон или проектируется новый шаблон (который может стать повторно используемым), задающий структуру передаваемой информации.

Для разработки программного комплекса в фонде платформы сформирована предметная область "Виртуальные интерактивные среды". Она состоит из следующих подразделов: концептуальные знания (используется для хранения онтологии), сервисы (содержит сервисы программного комплекса – графический редактор и интерпретатор ПВС,служебные ресурсы (предназначен для хранения вспомогательных информационных ресурсов, используемых сервисами), проекты виртуальных сред (содержит каталог моделей ПВС для решения различных задач)

4.4. Методы реализации программного комплекса

4.4.1. Структурный редактор для формирования логического представления модели

В соответствии с идеологией платформы IACPaaS все информационные ресурсы имеют единое унифицированное декларативное представление – в виде семантической сети. Основной особенностью информационных ресурсов

является их представление в виде пары информация – ссылка на метаинформацию. Метаинформация информационного ресурса является языком, в терминах которого формируется информация. Декларативные модели профессиональных сред также формируются как информационные ресурсы, для которых метаинформацией служит описанная во второй главе онтология ПВС.

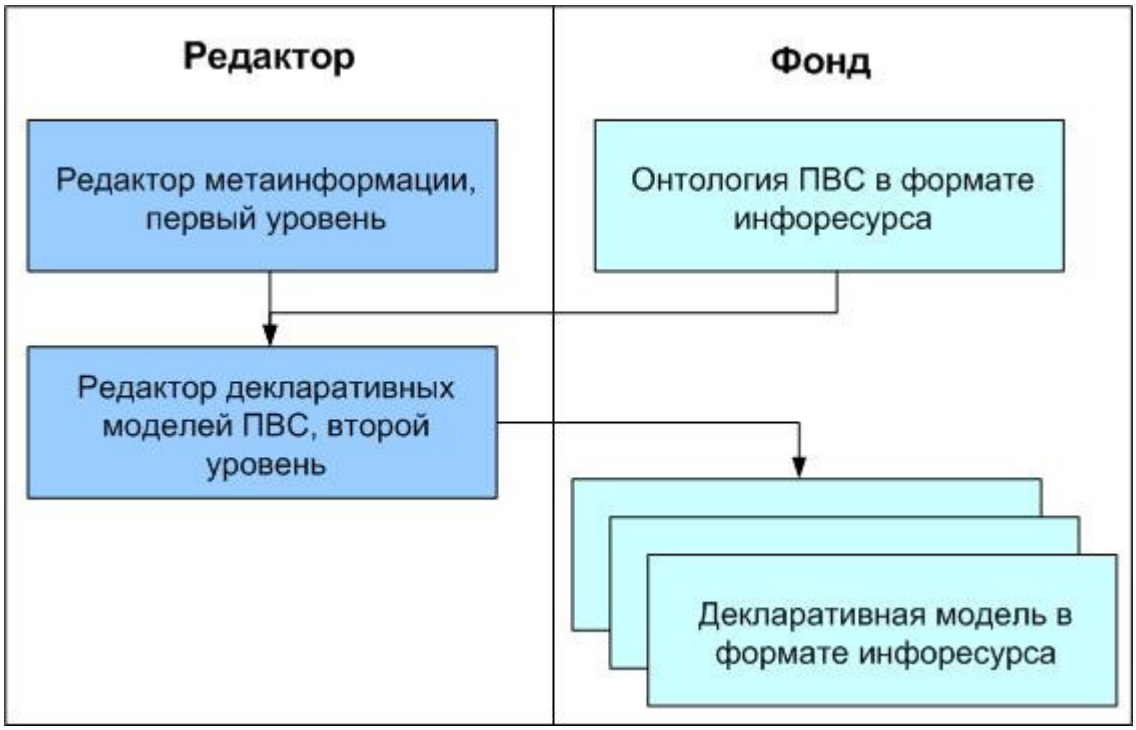


Рис. 4.4.1. Схема редактора инфоресурсов

Одним из компонентов (сервисов) платформы является редактор информационных ресурсов, используемый для формирования логического представления модели ПВС. Редактор является двухуровневым. Сначала формируется метаинформация, по которой автоматически создается редактор для создания информации. В разрабатываемом программном комплексе метаинформацией является онтология ПВС, информацией – модель ПВС. Таким образом, задача создания редактора декларативных моделей ПВС по онтологии сводится к разработке метаинформации на языке ИРУО первого уровня редактора IWE.

На рис. 4.4.1 представлена схема данного редактора.

Редактор обеспечивает создание, редактирование, удаление декларативной модели (информационного ресурса), осуществляет связь разработанной онтологии и создаваемой на ее основе декларативной модели, обеспечивая ее целостность.

4.4.2. Графический 3D-редактор для формирования презентационного представления модели

Для создания графического представления декларативных моделей в рамках диссертационной работы разработан редактор трехмерных сцен, являющийся одним из сервисов платформы. Данный редактор предоставляет веб-интерфейс и позволяет дополнить логическое представление модели ПВС (сформированное экспертом в структурном редакторе) его презентационным представлением.

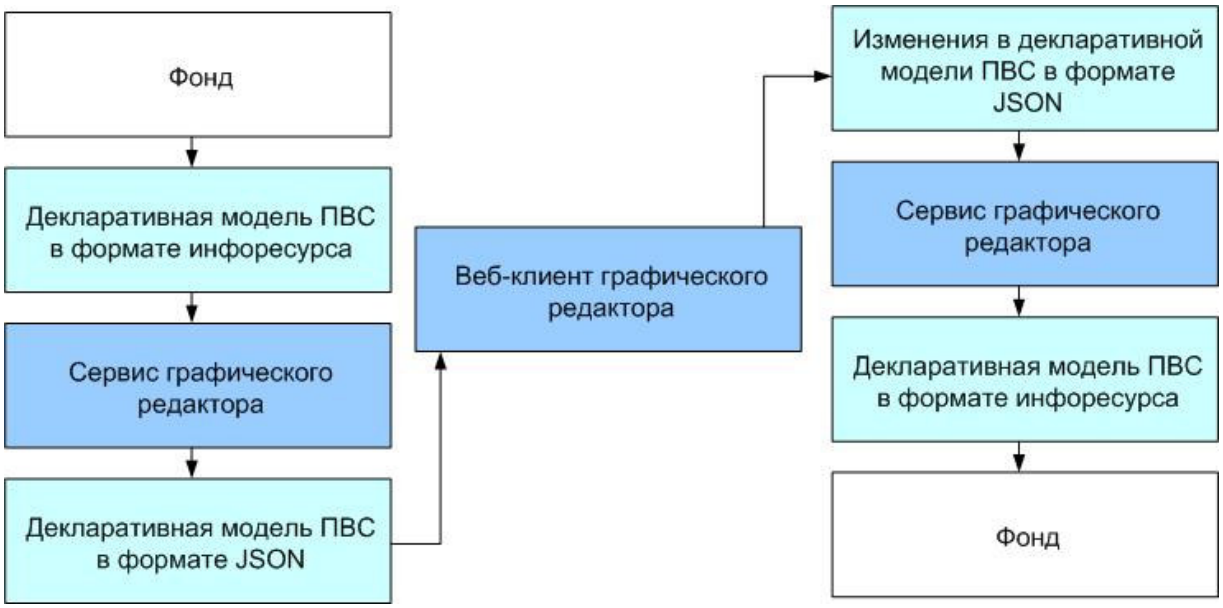


Рис. 4.4.2. Схема работы графического редактора

На рис. 4.4.2 представлена схема работы графического редактора.

Реализация серверной части графического редактора. Реализация серверной части графического редактора (сервиса) осуществлена на языке программирования Java с использованием мультиагентного подхода, поддерживаемого платформой IACPaaS.

Сервис графического редактора включает два агента: инициализирующий агент и обрабатывающий интерфейсный агент. Инициализирующий агент

производит инициализацию редактора как сервиса платформы и передает управление интерфейсному агенту с помощью инициализирующего сообщения. Интерфейсный агент отображает веб-клиент графического редактора и обрабатывает сообщения, исходящие от него.

Сообщения интерфейсному агенту (см. рис. 4.4.3) графического редактора содержат запросы на обработку информации, на обновление информации на сервере, на отправку и загрузку файлов на сервер платформы. Каждому сообщению сопоставлен соответствующий функциональный блок обработки (функция обработки) интерфейсного агента.

Сообщения интерфейсному агенту:

1) init. Единственное сообщение от инициализирующего агента. Приходит только один раз после запуска сервиса.
2) load. Сообщение, предназначенное для загрузки декларативной модели ПВС
3) newLib. Сообщение для создания новой библиотеки 3d-моделей
4) loadLibsList. Сообщение для загрузки списка доступных библиотек 3d-моделей.
5) loadLibModelsList. Сообщение для загрузки списка 3d-моделей библиотеки
6) download. Сообщение для загрузки файла ресурса с сервера
7) newFileUpload. Сообщение для загрузки нового файла на сервер
8) loadLibModel. Сообщение для загрузки 3d-модели из библиотеки
9) newModelUpload. Сообщение для загрузки новой модели на сервер
10) saveObject. Сообщение для сохранения объекта виртуальной сцены
11) saveScene. Сообщение для сохранения всей сцены

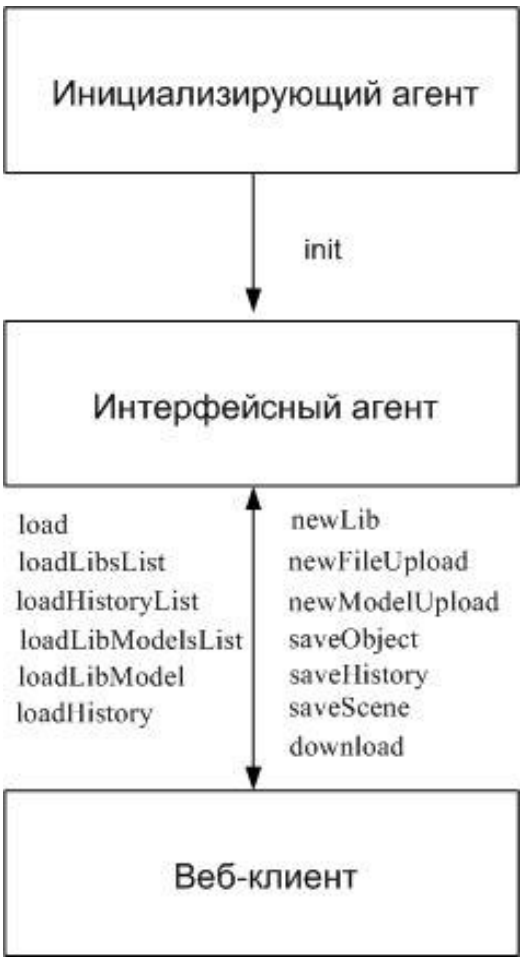


Рис. 4.4.3. Сообщения интерфейсному агенту графического редактора

Реализация сообщений

1) **init**. Результат обработки данного сообщения – это создание веб-клиента графического редактора и отображение его в качестве интерфейса сервиса. Для веб-клиента создается оконный интерфейс с помощью функции API платформы: uiShowWindowMessage.create(). В созданное окно помещается Flash-файл веб-клиента с помощью функции setInterface(). Flash-файл клиента устанавливается функцией flash() с нужными параметрами размеров и типа отображения.

2) **load**. Цель данного сообщения – отправить веб-клиенту декларативную модель в нужном формате. Для этого полученная с помощью функции getOutputs() декларативная модель ПВС, представленная в форме инфоресурса, преобразуется в формат JSON и возвращается веб-клиенту в качестве результата с помощью функций uiReturnInforesourceMessage() и setResult().

3) **newLib**. Цель данного сообщения – создать новую библиотеку в ресурсах сервиса. Для этого сначала обеспечивается доступ к ресурсам сервиса с помощью функций getOwns(), getGenerator(), generateFromAxiom(). С помощью функции nextByMeta() в дереве ресурсов находится нужная вершина и затем создается новая библиотека с помощью функции generateStraigt(). В качестве результата сообщения возвращается "ок", если все прошло успешно, и ничего не возвращается, если произошла какая-то ошибка.

4) **loadLibsList**. Результат данного сообщения – возвращение списка доступных библиотек. Из собственного инфоресурса извлекается вершина "библиотеки": data.nextByMeta("библиотеки"); затем в цикле по всем библиотекам: for (IConcept iter : conc.nextSetByMeta("библиотека")) формируем список библиотек в виде строки с разделителем ",".

5) **loadLibModelsList**. Результат данного сообщения – список с информацией о моделях библиотеки. С помощью функции getParam() извлекается название нужной библиотеки из параметров сообщения. С помощью функций nextByMeta() находится нужная библиотека. Затем в цикле производится обход всех моделей библиотеки: for (IConcept iter : conc.nextSetByMeta("модель")) для каждой модели добавляются ее параметры: тип модели, тип мэша, тип анимации, длина анимации, ключи анимации, имена анимаций; и добавляются в список в виде строки с разделителем ("|").

6) **download.** Результат данного сообщения – загрузка указанного медиа-файла. Из параметров сообщения (msg) извлекаются необходимые данные о файле с помощью функции getParam() из API платформы: название библиотеки: msg.getParam("libName"); название файла: msg.getParam("modelName"). В "собственном инфоресурсе" сервиса (получаемого с помощью функции getOwns) находится необходимый файл с помощью функций API платформы поиска по дереву инфоресурса: nextByMeta() и nextSetByMeta(). Найденный файл в формате бинарных данных BLOB прикрепляется к выходному сообщению с помощью функции API attachmentStream() и записывается в выходной поток класса OutputStream его методом write().

7) **newfileuploaded**. Задача данного сообщения – выполнить загрузку нового файла на сервер платформы. С помощью функции getFile() из сообщения извлекается файл, функцией getOwns() – доступ к собственным инфоресурсам сервиса. С помощью функции generateWithValue() сохраняем файл в виде бинарных данных BLOB: data.generateWithValue("файл", f.data). Если операции выполнились успешно веб-клиент получит ответное сообщение об успехе "Загружен файл <имя файла>".

8) **loadLibModel.** Результат данного сообщения – загрузка подробной информации о модели из библиотеки: имя, тип модели, тип сетки, флаг анимации, длина анимации, ключи анимации, имена анимаций. Из параметра сообщения извлекаются следующие данные: название библиотеки и название модели с помощью функции getParam(), с помощью функций nextByMeta() и nextSetByMeta() находится нужная модель и необходимые данные о модели. Итоговое ответное сообщение формируется как список найденных параметров модели в виде строки через разделители.

9) **newmodelupload**. Задача данного сообщения – загрузить все данные о конкретной модели: файл сетки (мэша), текстуры, иконки, а также сохранить всю информацию о данной модели: имя, тип модели, тип сетки, флаг анимации, длина анимации, ключи анимации, имена анимаций. С помощью функций getParam() из сообщения извлекаются все необходимые данные о модели. С помощью функции nextByMeta() в собственных ресурсах находится нужная вершина для модели. С помощью функций generateWithValue() сохраняются все данные о модели. Например, сохранение типа сетки: data.generateWithValue("meshType", meshType);

10) **saveObject**. Результат данного сообщения – сохранение в декларативной модели новых данных об атрибутах объекта. Данные об объекте приходят в формате JSON и считываются из строки сообщения методом: Json.read(objectStr). Функцией getGenerator() осуществляется доступ на изменение декларативной модели. С помощью функций nextByMeta() осуществляется поиск нужного объекта в дереве декларативной модели.

Функциями at(), getValue() из объекта Json выделяются параметры объекта: имя, тип, логические атрибуты, презентационные атрибуты, состояния. С помощью функции generateStraight() в атрибуты объекта записываются новые понятия. Если у объекта есть состояния, Json объект переводится в класс ассоциативного множества со строковыми ключами Set<String>. В цикле производится обход этого множества состояний и записываются в соответствующие атрибуты декларативной модели. Значения атрибутов записываются с помощью функций платформы getEditor().setValue().

11) **saveScene**. Результат данного сообщения – сохранение всей виртуальной сцены декларативной модели. Реализация данной функции осуществляется распарсиванием массива объектов сцены и цикличным вызовом функции сохранения каждого объекта.

Реализация веб-клиента редактора

Подготовка программного окружения

Большинство низкоуровневых задач по созданию трехмерной виртуальной среды берет на себя выбранный 3d-движок: рендеринг, отсечение, сортировка, текстурирование и другие. Для использования движка его модули подключаются к проекту, инициализируются и применяются необходимые классы: Object3D – общий класс трехмерных объектов сцены, View – класс окна отображения трехмерной сцены, Resource – класс ресурсов, используемых движком и видеокартой компьютера, Camera3D – класс, используемый для создания камеры, перевода мировых координат сцены и локальные и экранные; SimpleObjectController – котроллер управления камерой, Mesh – класс для создания трехмерных «сеток» (физической структуры) объектов, Material – класс для создания отображения объектов сцены при помощи покрытия сеток, DirectionalLight – класс для создания направленного источника света, DirectionalLightShadow – класс для создания теней от объектов при освещении направленным источником света; и другие классы.

Создание виртуальной сцены

Создание виртуальной сцены включает несколько этапов:

- Создается пустая сцена с помощью создания первого пустого трехмерного объекта Object3D.
- Создается первичная камера Camera3D, через которую пользователь "видит" виртуальную сцену.
- Создается котроллер управления этой камерой SimpleObjectController.
- Создается окно отображения виртуальной сцены: View с необходимыми размерами ширины и высоты и присоединяется к камере: addChild(camera.view);
- Создается базовый источник света DirectionalLight, который обеспечивает базовое затенение объектов. К нему добавляются тени: DirectionalLightShadow.

После выполнения данных операций виртуальная среда может считаться подготовленной к формированию трехмерной сцены. Программа имитации виртуальной среды работает до тех пор, пока ее не закрыл пользователь. Во время выполнения трехмерная сцена может меняться, пользователь может взаимодействовать с виртуальной средой.

Для организации непрерывного рендеринга создается бесконечный цикл, который повторяется каждые 30 миллисекунд: onEnterFrame(). Внутри цикла происходят постоянные обновления: checkCollisions() – проверка коллизий и взаимодействий пользователя со средой, sceneManager.update() – обновление сцены; myInterface.update() – обновление интерфейса; camera.render(stage3D) – новый рендеринг сцены по отношению к текущей камере.

Реализация классов и функций редактора

На рис. 4.4.4 представлена диаграмма классов, реализующих необходимые функции графического редактора.

Все классы программы разделены на группы: группа классов сцены, группа классов инструментов, группа интерфейсных классов, группа утилит. Класс Main является входной точкой программы, инициализирующий движок рендеринга и остальные классы.

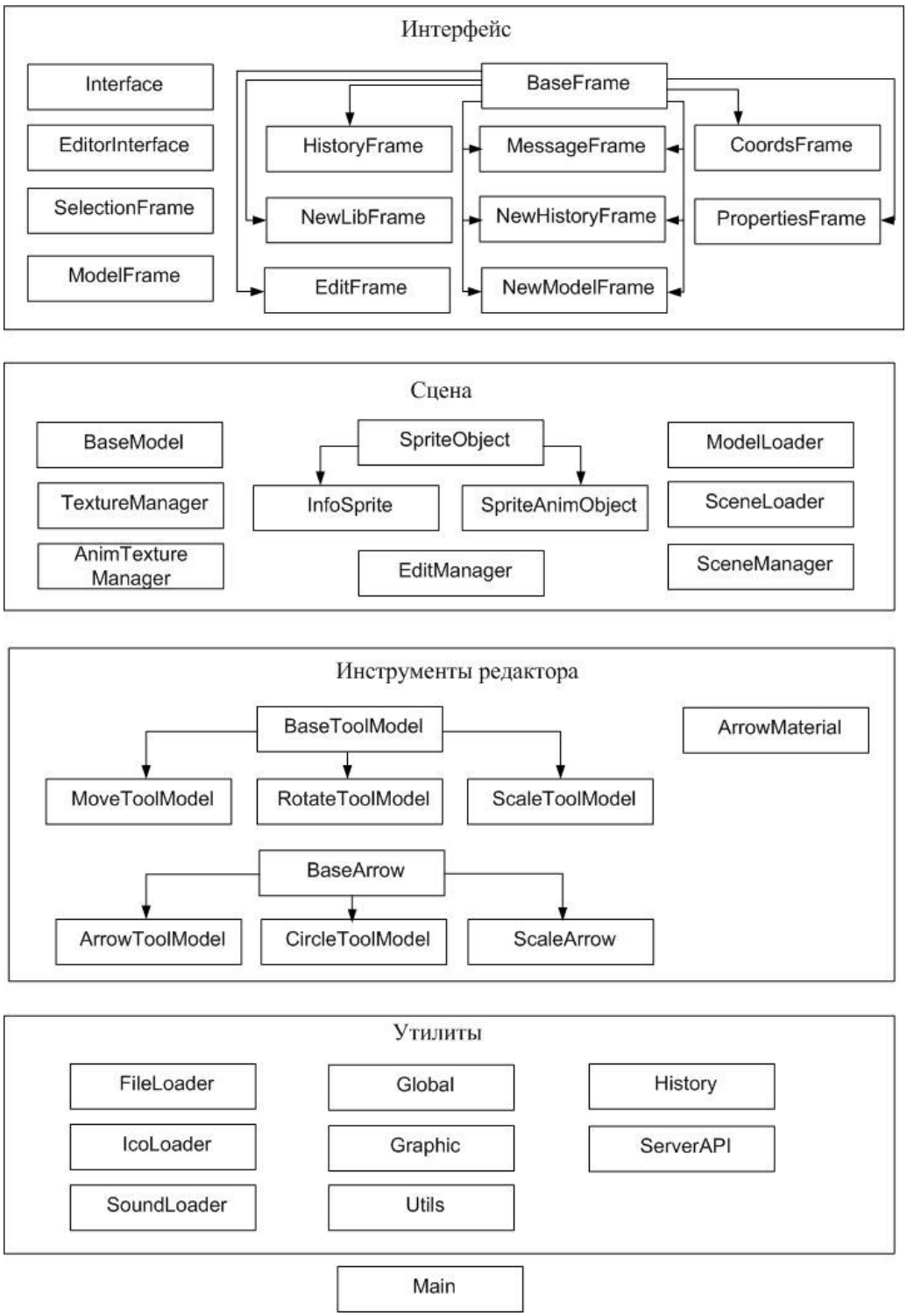


Рис. 4.4.4. Диаграмма классов графического редактора

Описание реализации классов и их функций клиента графического редактора приведено в Приложении 1.

Модель интерпретации была описана в главе 3. Соответствие (реализация) абстрактных функций через описанные выше классы и их методы выглядит следующим образом:

создать_объект (имя, описание) - SceneLoader.loadNoad(),.

загрузить_модель (модель) - SceneLoader.loadModel(), ModelLoader.loadMesh()

загрузить_текстуры (текстуры) - ModelLoader.loadTexture().

разместить_объект (x,y,z) - ModelLoader.initData().

повернуть_объект (x,y,z) - ModelLoader.initData().

масштабировать_объект (x,y,z) - ModelLoader.initData().

создать_источник_света (цвет, сила, тень, тип) - SceneLoader.createOmniLight()

создать_камеру () – SceneLoader.createCamera().

JSON

Сторонние (внешние) по отношению к платформе программные модули (веб-клиенты) не обладают ни доступом к информационным ресурсам, ни API к ним с платформы. Информационные ресурсы платформы хранятся в формате семантических сетей, но могут быть представлены (средствами конвертации, предоставляемыми платформой IACPaaS) в формате JSON, общепринятом в Интернете (наряду с XML) формате описания и передачи сложно-структурированных текстовых данных.

Интерфейс графического редактора

Функции редактора:

- создание, отображение и управление виртуальной сценой
- создание, отображение, размещение, трансформация объектов декларативной модели
- загрузка, сохранение (в декларативной модели) ресурсов виртуальной сцены: трехмерных моделей объектов, текстур, иконок, анимаций;
- использование и отображение повторно используемых ресурсов
- сохранение виртуальной сцены в декларативную модель

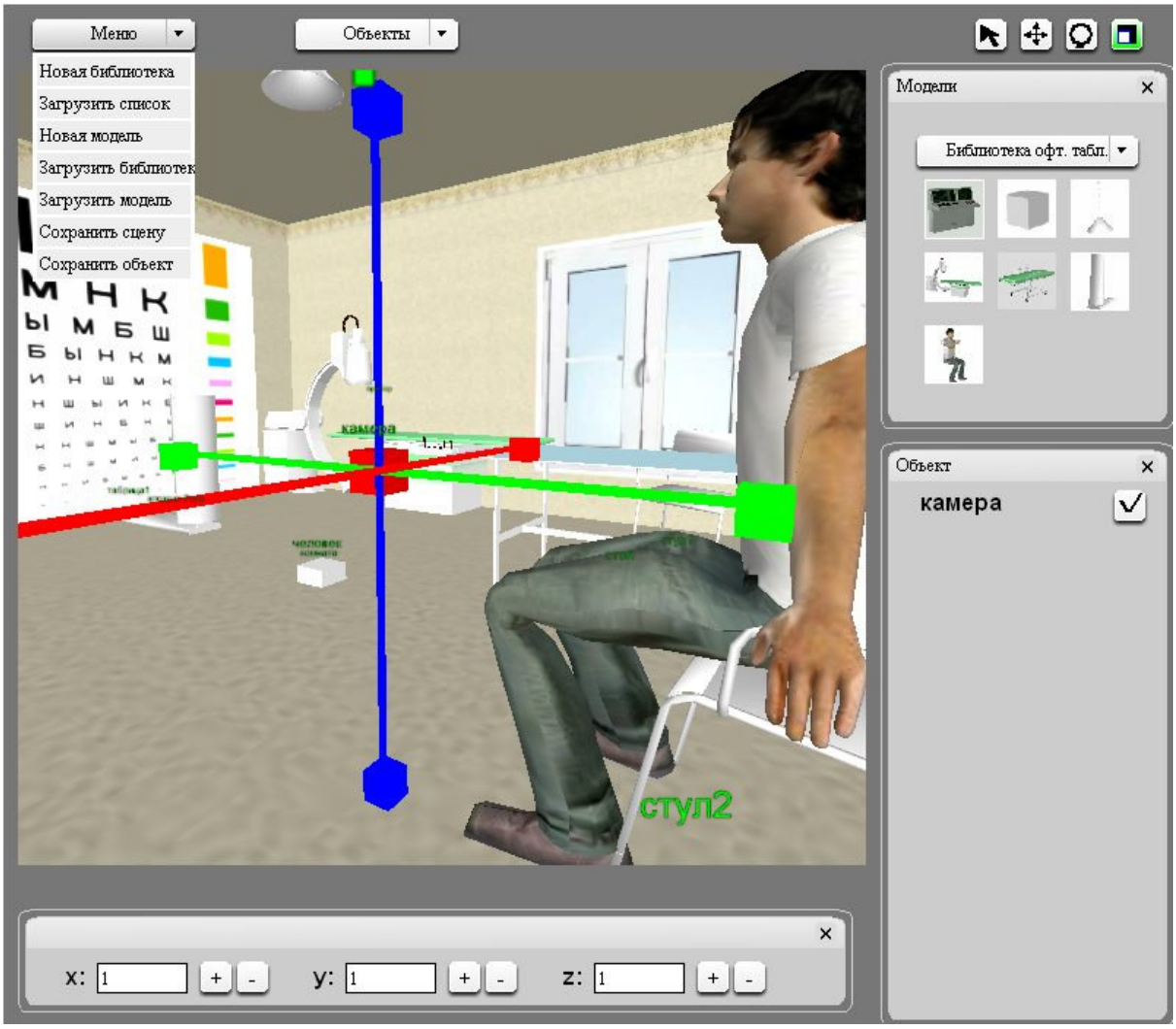


Рис. 4.4.5. Графический редактор трехмерных сцен

На рис. 4.4.5 представлен интерфес графического редактора.

4.4.3. Интерпретатор

Профессиональные виртуальные среды разрабатываются на платформе IACPaaS с помощью сервисов платформы и становятся также одними из ее сервисов. Для создания сервиса из разработанной декларативной модели необходима дополнительная система – специальный сервис, запускающий и обрабатывающий ПВС, т.е. необходим сервис-интерпретатор декларативных моделей [41, 107-108].

Схема работы интерпретатора

На рис. 4.4.6 представлена схема работы интерпретатора.

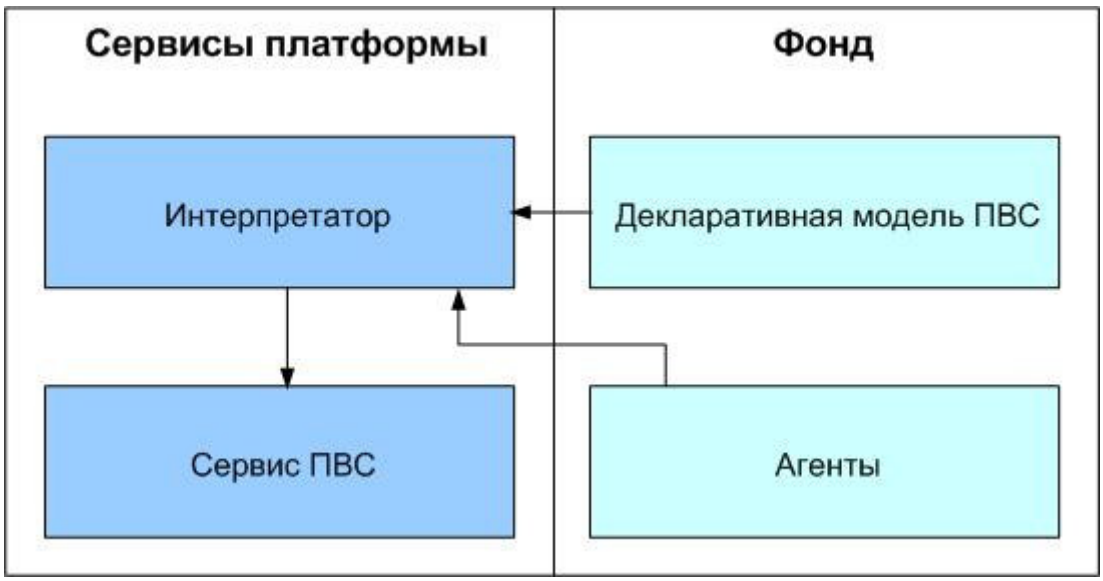


Рис.4.4.6. Схема сервиса интерпретатора

На рис. 4.4.7 представлена схема работы веб-клиента интерпретатора.

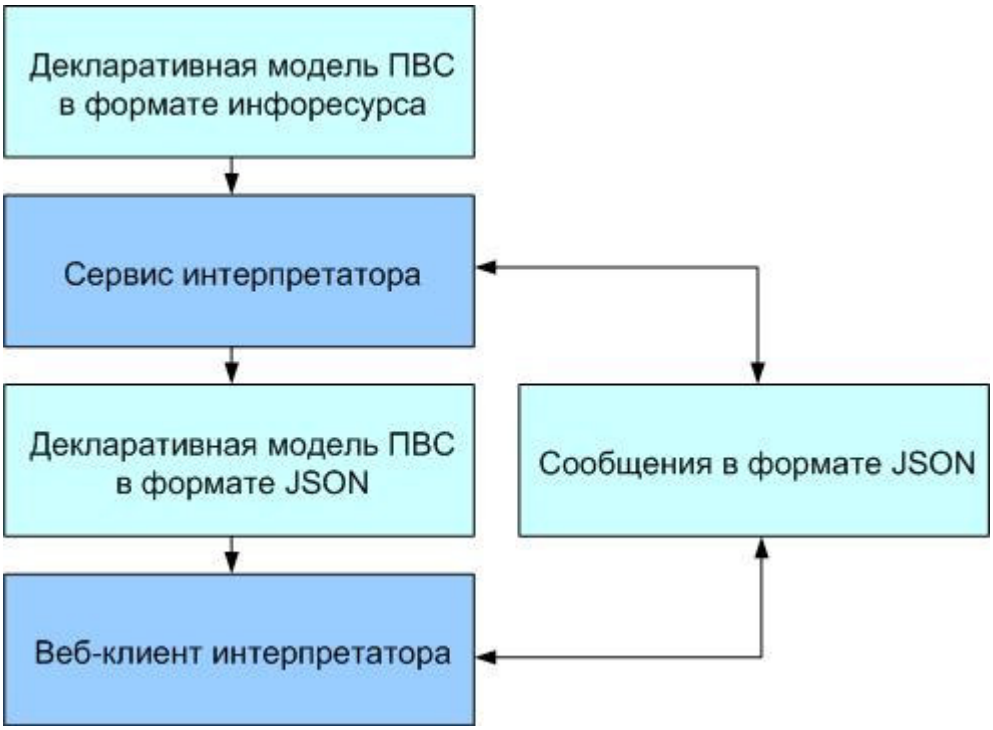


Рис. 4.4.7. Веб-клиент интерпретатора

Реализация серверной части сервиса интерпретатора

Реализация серверной части сервиса интерпретатора профессиональных виртуальных сред осуществлена аналогично сервису "Графический редактор" на языке программирования Java с использованием мультиагентного подхода, поддерживаемого платформой IACPaaS.

Сервис интерпретатора включает два агента: инициализирующий агент и обрабатывающий интерфейсный агент. Инициализирующий агент производит инициализацию интерпретатора как сервиса платформы и передает управление интерфейсному агенту с помощью инициализирующего сообщения. Интерфейсный агент отображает веб-клиент интерпретатора и обрабатывает сообщения, исходящие от него.

Сообщения интерфейсному агенту интерпретатора содержат запросы на обработку информации, на обновление информации на сервере, на отправку и загрузку файлов на сервер платформы. Каждому сообщению сопоставлен соответствующий функциональный блок обработки (функция обработки) интерфейсного агента.

Сообщения интерфейсному агенту:

1) init. Единственное сообщение от инициализирующего агента. Приходит только один раз после запуска сервиса.

2) load. Сообщение, предназначенное для загрузки декларативной модели ПВС

3) download. Сообщение для загрузки файла ресурса с сервера

4) loadLibModel. Сообщение для загрузки 3d-модели из библиотеки

5) action. Сообщение для обработки действия пользователя

6) runProduction(ProcessVRMessage msg, InterpreterVRInterfaceAgentImpl.ProcessVRMessageResultCreator rc). Отдельный блок продукции интерфейсного агента для обработки действия пользователя.

7) initExercise. Сообщение для инициализации упражнения, выбранного пользователем

Реализация сообщений

1) **init.** Результат обработки данного сообщения – это создание веб-клиента интерпретатора и отображение его в качестве интерфейса сервиса. Для веб-клиента создается оконный интерфейс с помощью функции АПИ платформы: uiShowWindowMessage.create(). В созданное окно помещается Flash-файл веб-

клиента с помощью функции setInterface(). Flash-файл клиента устанавливается функцией flash() с нужными параметрами размеров и типа отображения.

2) **load.** Результат данного сообщения – отправка веб-клиенту декларативной модели в формате JSON. Для этого полученная с помощью функции getOutputs() декларативная модель ПВС, представленная в форме инфоресурса, преобразуется в формат JSON и возвращается веб-клиенту в качестве результата с помощью функций uiReturnInforesourceMessage() и setResult().

3) **download.** Результат данного сообщения – загрузка указанного медиа-файла. Реализуется аналогично сообщению для графического редактора.

4) **loadLibModel.** Результат данного сообщения – загрузка подробной информации о модели из библиотеки: имя, тип модели, тип сетки, флаг анимации, длина анимации, ключи анимации, имена анимаций. Реализуется данное сообщение аналогично сообщению из сервиса графического редактора.

5) **action**. Результат данного сообщения – выполненная обработка действия пользователя. Извлекаются параметры сообщения, имя действия и имя объекта (для интерактивных действий) с помощью функции getParam(). С помощью функций поиска по дереву инфоресурса из API платформы находится нужное действие в декларативной модели. Если в декларативной модели задана дополнительная обработка действия внешними агентами, то в цикле формируются и отправляются сообщения соответствующим агентам с помощью функции rc.processVRMessage.create(<имя агента>). Если в модели не задано ни одной дополнительной внешней обработки, тогда отправляется сообщение интерфейсному агенту (т.е. самому себе) о продолжении обработки с помощью функции self(): rc.processVRMessage.create(self()). В каждое сообщение (как самому себе, так и другим внешним агентам) добавляются все параметры исходного сообщения, имя действия, имя объекта (если есть) и текущая версия полной декларативной модели. Таким образом, в любом случае выполнение исходного сообщения прекращается отправкой нового сообщения сервису и ожиданием и получением интерфейсным агентом дальнейшего

сообщения для обработки действия с помощью блока продукции runProduction(ProcessVRMessage msg, InterpreterVRInterfaceAgentImpl.ProcessVRMessageResultCreator rc)

6) runProduction(ProcessVRMessage msg, InterpreterVRInterfaceAgentImpl.ProcessVRMessageResultCreator rc)

Данный блок продукции выполняется в двух случаях: сразу же после выполнения сообщения "action", когда нет внешних агентов обработки; после выполнения внешних агентов: после того как они вышлют интерфейсному агенту соответствующее сообщение.

В данном блоке продукции выполняются "стандартные" обработчики действия: а) изменение состояний объектов (если заданы), б) получение оценки действия (если задано), в) проверка действия по сценарию (если сценарий задан).

а) Изменение состояний объектов. С помощью функции nextSetByMeta() в цикле выполняется обработка всех заданных для действия состояний объектов:

for (IConcept iter : cAct.nextSetByMeta("изменение состояния объекта"))

С помощью функции gotoByMeta() определяется ссылка на состояние объекта:

tempConc = iter.gotoByMeta("ссылка на состояние изменяемого объекта");

if (tempConc == null) iter.gotoByMeta("ссылка на состояние составного объекта");

С помощью функции getIncomingRelations() из ссылки извлекается имя объекта и название состояния. С помощью функции nextByMeta() в декларативной модели осуществляется поиск соответствующего состояния объекта. Для обновления состояния объекта выполняется обновление его презентационных атрибутов с помощью функций getConceptNameOrValue(),getEditor() и setValue(); и логических атрибутов с помощью функций поиска выходящих отношений getOutcomingRelations() и функций обновления данных getEditor().setValue().

б) Операция получения оценки действия выполняется с помощью цикла по всем "оценкам параметров", заданных для действия:

for (IConcept iter : valueResultParams.nextSetByMeta("оценка параметров"))

Затем в цикле проверяются атрибуты объектов:

for (IConcept param : iter.nextSetByMeta("значение параметра"))

С помощью функций gotoByMeta() извлекаются данные "ожидаемого значения" и полученного "значения") и затем сравниваются по заданной операции сравнения. Если хотя бы одна проверка параметров объектов не прошла успешно выполняется переход к следующему варианту параметров оценок. Если все проверки атрибутов объектов прошли успешно, выполняется аналогичная проверка параметров, выполненных внешними агентами. Если проверки всех параметров оценки прошли успешно, то оценка, соответствующая проверенным параметрам добавляется в список возможных оценок и далее продолжается проверка других оценок параметров. Если в результате проверки всех возможных оценок нашлось несколько удовлетворяющих условиям оценок, итоговая оценка действия выбирается случайным образом. Если не найдена ни одна оценка, то в качестве ответа возвращается значение "-1".

в) В независимости от изменений состояний объектов и получения оценки действия после этих операций выполняется проверка действия со сценарием (если таковой присутствует в модели). С помощью функции gotoByMeta() находится текущий этап и вершина в сценарии. Аналогично находится соответствующее вершине действие. Ожидаемое действие ("действие в вершине") сравнивается с выполненным действием пользователя. Если проверка прошла неуспешно, то возвращается ответ "неверно" с комментарием "выполнено не то действие, ожидалось действие <действие в вершине>". Если действия совпали, тогда выполняется проверка соответствия параметров полученных и ожидаемых. Аналогично с помощью функции gotoByMeta() извлекаются "ожидаемое значение" и полученное "значение" и сравниваются. Если проверка параметров прошла неуспешно, то возвращается ответ "неверно"

с комментарием "Действие выполнено неверно: ожидался параметр <ожидаемое значение>". Если все проверки параметров прошли успешно, то выполняется поиск возможных переходов в следующую вершину сценария. Для этого выполняется цикл по всем дугам, исходящим от текущей вершины сценария:

for (IConcept arc : arcs.nextSetByMeta("дуга"))

Для каждой дуги извлекается пара вершин:

cVertex1 = arc.gotoByMeta("вершина, из которой выходит дуга");

cVertex2 = arc.gotoByMeta("вершина, в которую входит дуга");

и заданный для дуги переход:

cTrans = arc.gotoByMeta("переход");

Выполняется проверка типа перехода с помощью функции gotoByMeta().

Если переход "по оценке":

if (cTrans.gotoByMeta("по оценке") != null)

то извлекаем соответствующую оценку:

resValue = getObjectAttrStateName(conc, "оценка");

и выполняется проверку с полученной ранее оценкой.

Если переход "по условию":

if (cTrans.gotoByMeta("по условию") != null)

тогда аналогично проверке параметров при получении оценки выполняется проверка заданных для перехода значений атрибутов и текущих в декларативной модели.

Если все прошло успешно, то обновляется значение "текущая вершина" для сценария во временной копии декларативной модели с помощью функций getEditor().setValue().

В итоге возвращается ответ в виде строки, содержащий оценку действия, комментарий оценки.

7) **initExercise.** Результат данного сообщения – инициализировать логические атрибуты объектов. Для этого выполняется поиск по всем объектам декларативной модели в цикле и по всем их атрибутам. Если у атрибута

объекта задано несколько возможных значений, то устанавливается одно из них случайным образом. Если возможные значения заданы перечислением через запятую ",", то данное перечисление (строка) разбивается на массив возможных значений с помощью стандартной функции split(). Если возможные значения заданы множеством через "дефис", то возможные целочисленные значения выбираются из заданного диапазона.

Реализация веб-клиента интерпретатора

Веб-клиент интерпретатора реализован аналогично веб-клиенту графического редактора создания ПВС и представляет собой веб-клиент на основе Flash-модуля.

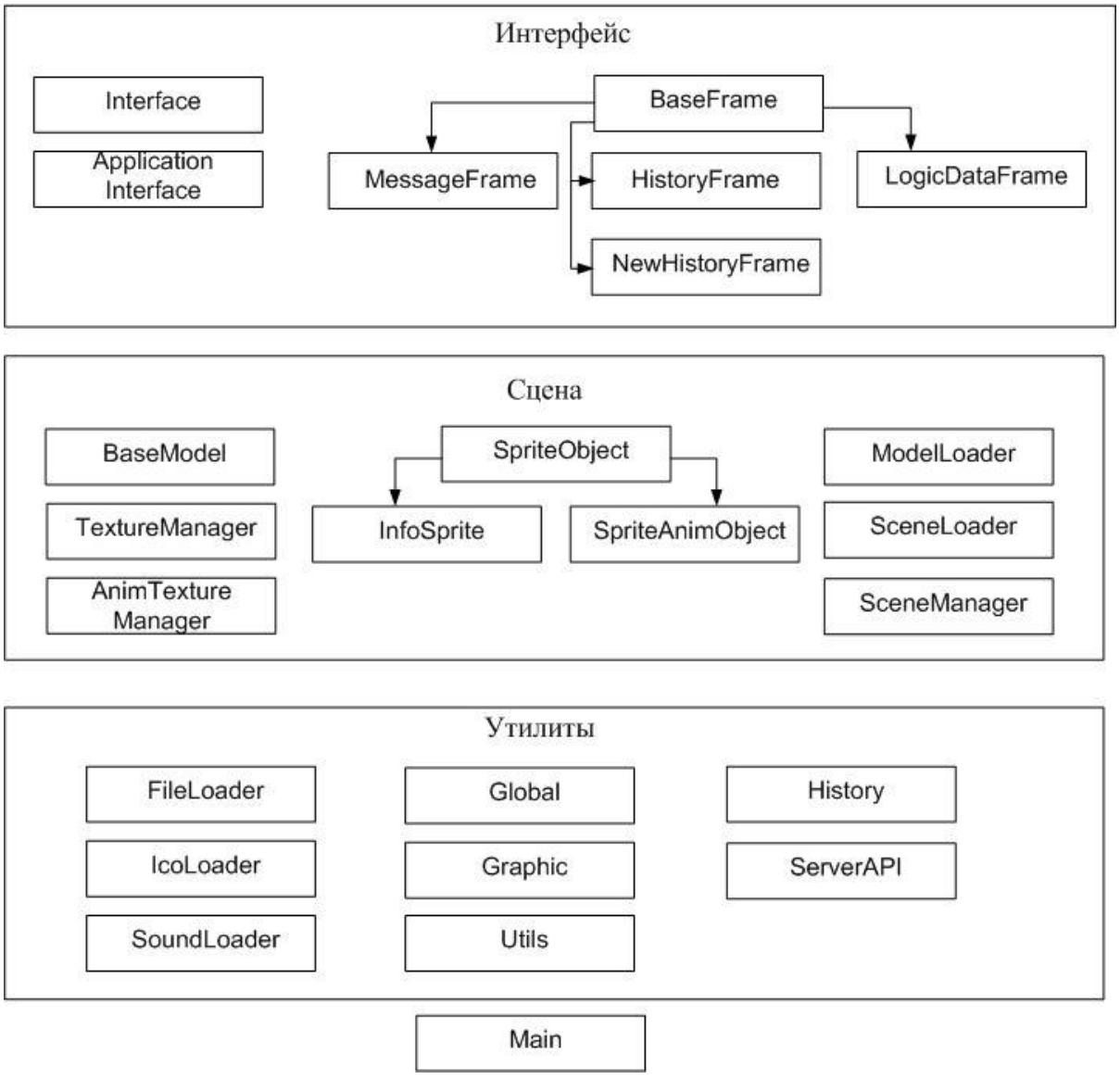


Рис. 4.4.8. Диаграмма классов интерпретатора

Аналогично реализовано программное окружение, создание виртуальной сцены и непрерывного рендеринга. Декларативная модель ПВС также как и в редакторе отправляется веб-клиенту интерпретатора в формате JSON.

Реализация классов и функций редактора

На рис. 4.4.8 представлена диаграмма классов, реализующих необходимые функции интерпретатора.

Все классы программы разделены на группы: группа классов сцены, группа классов инструментов, группа интерфейсных классов, группа утилит. Класс Main является входной точкой программы, инициализирующий движок рендеринга и остальные классы.

Описание реализации классов и их функций клиента графического редактора приведено в Приложении 1.

Интерфейс интерпретатора

Основные функции веб-клиента интерпретатора ПВС:

- создание, отображение и управление виртуальной сценой
- генерация ПВС по декларативной модели

Рис. 4.4.9. Интерфейс веб-клиента интерпретатора ПВС

• загрузка ресурсов виртуальной сцены: трехмерных моделей объектов, текстур, иконок, анимаций

• интерактивное взаимодействие с объектами сцены

• инициализация параметров ПВС

• сохранение и загрузка анимационных роликов

• отображение объяснений и результатов действий пользователя

На рис. 4.4.9. представлен интерфейс веб-клиента ПВС.

4.5. Сравнительные характеристики программного комплекса

Сравнение разработанного программного комплекса с аналогичными решениями приводится по следующим критериям (затем оформленного в виде таблицы 4.5.1):

• Настройка и развертывание. Программный комплекс разработан в рамках облачной платформы IACPaaS и доступен пользователем через Интернет. Это означает отсутствие необходимости сложной настройки и развертывания системы, в отличие от большинства средств разработки ПВС.

• Сопровождение и управление. Применение облачной платформы и работа инструментальных средств как веб-сервисов позволяет разработчикам сопровождать разработанные виртуальные среды удаленно, где бы они не находились. Применение декларативного подхода позволяет вести разработку, постоянно изменяя и отлаживая модель виртуальной среды, без необходимости ее перекомпиляции и загрузки на сервер. Другие системы имеют высокую трудоемкость сопровождения как из-за традиционного программного обеспечения (не веб-сервисов), так и из-за необходимости перекомпиляции и сложности обновления программного обеспечения у пользователей.

• Включение экспертов предметной области в процесс разработки. Созданный программный комплекс ориентирован на привлечение к работе над ПВС экспертов предметной области, которые способны без помощи инженеров по знаниям и программистов определить все логическое наполнение ПВС, в отличие от большинства других систем, в которых эксперты выступают только консультантами.

• Разделение работ между специалистами. Разработанный программный комплекс рассчитан на разделение разработчиков и, главным образом, ориентирован на экспертов предметной области и дизайнеров. В большинстве же других систем предполагается ориентированность на высококвалифицированных программистов.

• Доступность через Интернет. Разработанный программный комплекс доступен всем (как разработчикам, так и пользователям) через Интернет через облачную платформу IACPaaS. Большинство других систем реализованы как традиционные настольные программные системы, что означает сложность доступа к программам.

• Использование технологий современной компьютерной графики. Разработанный программный комплекс обладает высоким уровнем трехмерной компьютерной графики, не уступающим по качеству сложным настольным программным системам, использующим технологии DirectX и OpenGL.

Таблица 4.5.1. Сравнение подходов к разработке профессиональных виртуальных сред

	Программный комплекс для разработки профессиональных виртуальных сред	**Средства разработки виртуальных сред общего назначения**
Настройка и развертывание	Виртуальные среды работают как сервисы платформы IACPaaS. Нет необходимости что-то настраивать и устанавливать	Виртуальные среды представлены отдельными системами или exe-файлами, требующими настройки и отдельного развертывания у пользователей
Сопровождение и управление	Облачность платформы и интерпретация виртуальных сред обеспечивают простое сопровождение и управление	Любое изменение виртуальных сред требует перекомпиляции и переустановки системы у пользователей
Включение экспертов в процесс разработки	Эксперты предметной области принимают непосредственное участие в разработке, создавая декларативную модель на основе экспертных знаний без непосредственной помощи программиста	Эксперты предметной области могут выступать только консультантами для программистов

Разделение работ между специалистами	Разделение разработки между экспертами и дизайнерами обеспечивается соответствующим разделением онтологии на два уровня. Экспертам для работы предлагается структурный редактор. Дизайнерам - графический редактор.	Эксперты могут работать только вместе с другими специалистами. Дизайнеры не могут сформировать графическое представление сцены без участия программиста, который строит сцену программным способом из разработанных дизайнером моделей.
Доступность через Интернет	Облачность платформы обеспечивает доступ к профессиональным виртуальным средам через Интернет как веб-сервисам	Реализация доступа через Интернет требует дополнительной работы с Интернет-серверами, соответствующими скриптами, сессиями и т.п.
Использование технологий современной компьютерной графики	Технология Flash, положенная в основу клиентского модуля графического редактора и интерпретатора, обеспечивает вывод трехмерной графики через видео-карту и аппаратное ускорение, что обеспечивает современный уровень компьютерной графики	Использование современных графических библиотек на базе DirectX, OpenGL обеспечивает высокое качество компьютерной графики

4.6. Выводы

В результате выполнения четвертой задачи диссертационной работы разработаны методы реализации инструментального комплекса. Реализован интерпретатор, клиентский и серверный модули инструментария. Разработан 3d-редактор для создания трехмерных сцен виртуальных сред. Реализованный инструментальный комплекс работает как облачный сервис на платформе IACPaaS в соответствии со всеми предложенными принципами и предъявленными требованиями.

ГЛАВА 5. ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОФЕССИОНАЛЬНОЙ ВИРТУАЛЬНОЙ СРЕДЫ

В данной главе представлена технология разработки и использования профессиональных виртуальных сред (ПВС) с помощью интегрированного программного комплекса, а также приводятся примеры ПВС, реализованные с его помощью.

5.1. Технология разработки модели

Разработка модели профессиональной виртуальной среды включает несколько этапов [39]:

1. Разработка логического представления декларативной модели ПВС экспертами предметной области.

2. Разработка презентационного (графического) представления модели ПВС дизайнерами.

3. Технология разработки агентов программистами (данный этап не является обязательным, однако, если предоставляемого функционала оказывается недостаточно, его функционал может быть расширен дополнительными функциями).

Технология разработки логического представления декларативной модели

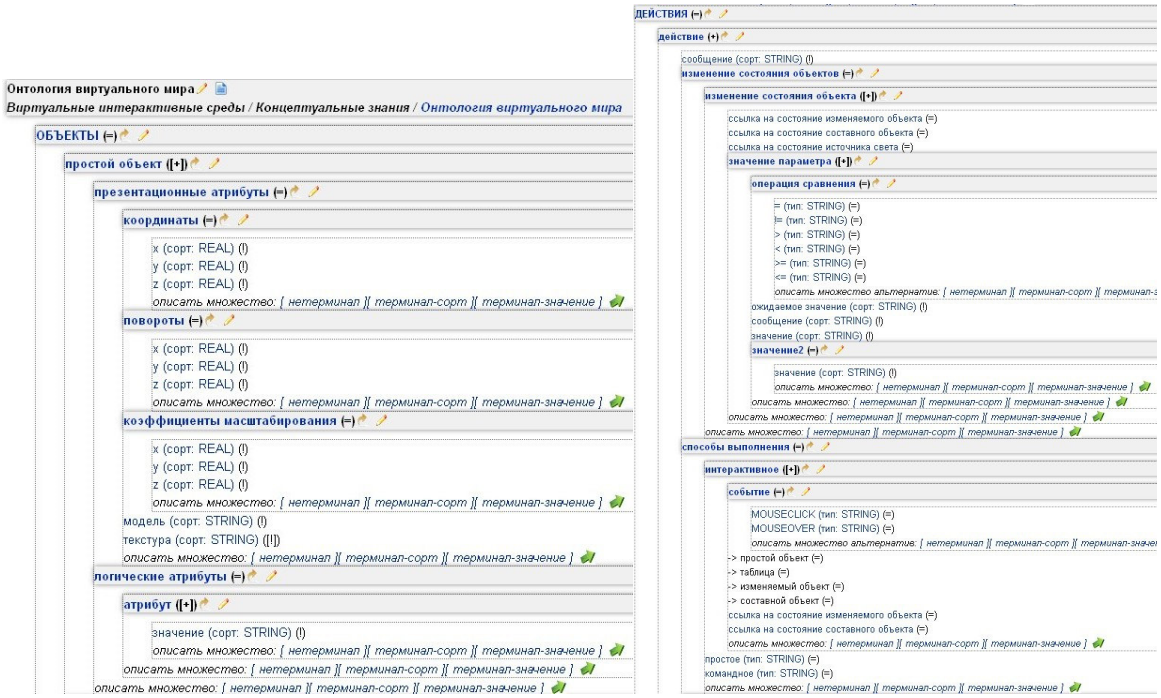
Технология разработки декларативной модели состоит из следующих шагов:

1. Создание информационного ресурса декларативной модели.
2. Формирование объектов.
3. Формирование действий.
4. Формирование сценария.

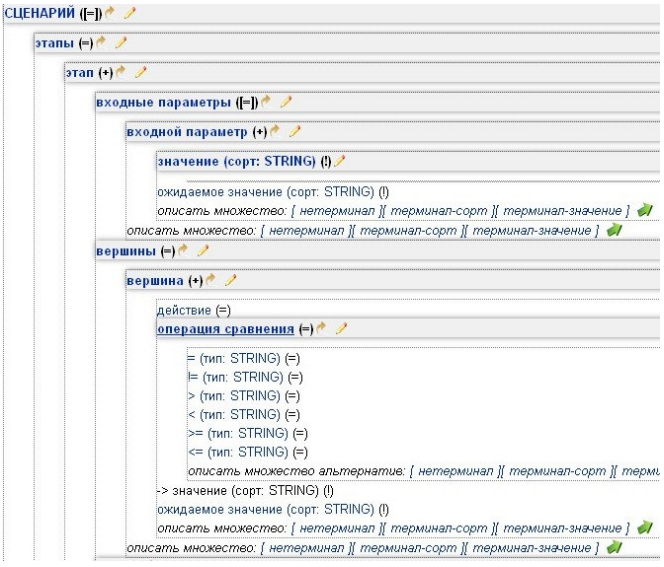
Создание информационного ресурса декларативной модели

Для создания информационного ресурса декларативной модели эксперт через административную систему платформы IACPaaS подает заявку на создание нового информационного ресурса, после создания которого приступает к формированию трех составляющих декларативной модели:

объектов, действий, сценария. Формирование модели и ее последующая модификация осуществляется с помощью редактора ИРУО, который является составным компонентом платформы IACPaaS. Формирование модели ПВС осуществляется в терминах онтологии ПВС, разработанной в рамках диссертационной работы.



а) б)



в)

Рис. 5.1.1.Структурный редактор. Примеры

Онтология ПВС поступает на вход редактору ИРУО, пользователь (эксперт) конкретизируя термины онтологии, получает логическое

представление конкретной модели ПВС. На рис. 5.1.1 (а-в) приведены скриншоты интерфейса редактора для формирования логического представления модели ПВС.

Формирование декларативной модели далее идет в следующем порядке: объекты, действия, сценарий. Объекты сцены являются независимыми сущностями. Действия зависят от объектов, ссылаются на них. Сценарий зависит и от объектов, и от действий.

Технология разработки презентационного представления модели

Разработка презентационного представления модели ПВС происходит после того, как эксперт выполнил формирование ее логического представления.

Презентационное представление сцены формирует дизайнер с помощью сервиса "Графический редактор трехмерных сцен", разработанного в рамках диссертационной работы.

В данный редактор загружается сформированное экспертом логическое представление модели ПВС. С помощью редактора в визуальном интерактивном режиме дизайнер определяет презентационные атрибуты объектов сцены. Таким образом, сервис "Графический редактор трехмерных сцен" имеет в качестве входного ресурса декларативную модель виртуальной среды, а в качестве выходного – полную модель ПВС, содержащую логическое и презентационное представление модели ПВС.

Сформированная экспертом декларативная модель виртуальной среды, впервые загруженная в графический редактор трехмерных сцен, т.е. еще без какого-либо презентационного наполнения, отображается в редакторе в виде набора выстроенных в линейку кубов, подписанных именами тех объектов, которые они представляют (см. рис. 5.1.2).

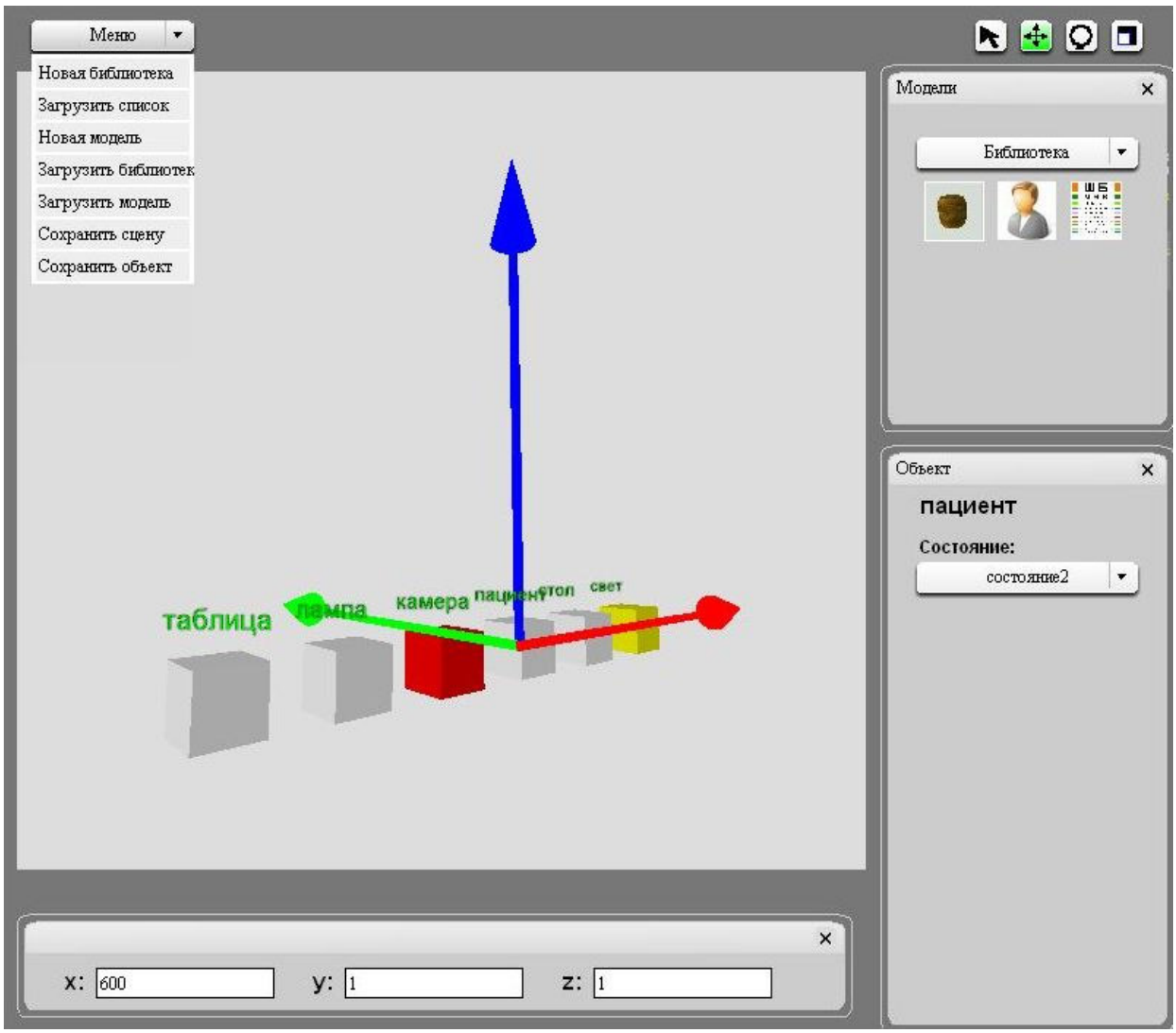


Рис. 5.1.2. Редактор. Инициализация сцены

На рисунке 5.1.2 представлен редактор в момент первой загрузки декларативной модели с логическим описанием сцены.

Для задания презентационных атрибутов дизайнеру необходимо определить: трехмерная модель объекта, текстура, координаты, повороты, коэффициенты масштабирования, анимации.

Для определения ***трехмерной модели*** некоторого объекта дизайнер выбирает подходящую модель в одной из имеющихся в редакторе библиотек (см. рис. 5.1.3).

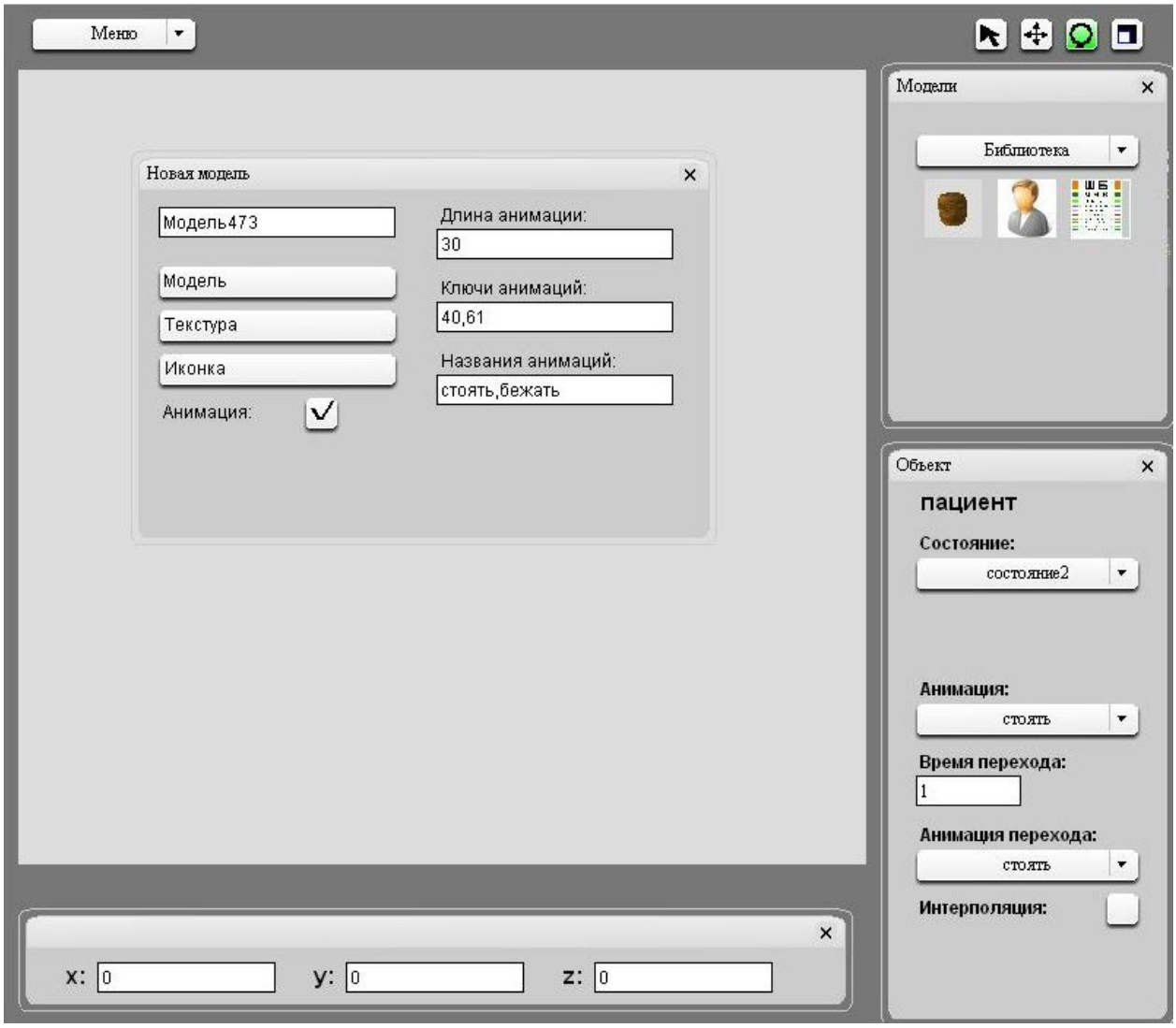


Рис. 5.1.3. Выбор трехмерной модели для объекта

Если требуемой модели в библиотеках нет, дизайнер может загрузить нужную модель из внешнего источника и поместить ее в одну из существующих библиотек, либо в новую библиотеку.

Для загрузки новой модели дизайнеру необходимо указать ее имя, источник файла 3d-модели (мэша), источник файла текстуры для этой модели и источник файла иконки (небольшой картинки) этой модели для отображения в интерфейсе библиотеки редактора (см. рис.5.1.4).

Позиционирование объекта

Позиционирование объекта на виртуальной сцене осуществляется в визуальном режиме с помощью направленных по координатным осям стрелкам, привязанным к выбранному объекту.

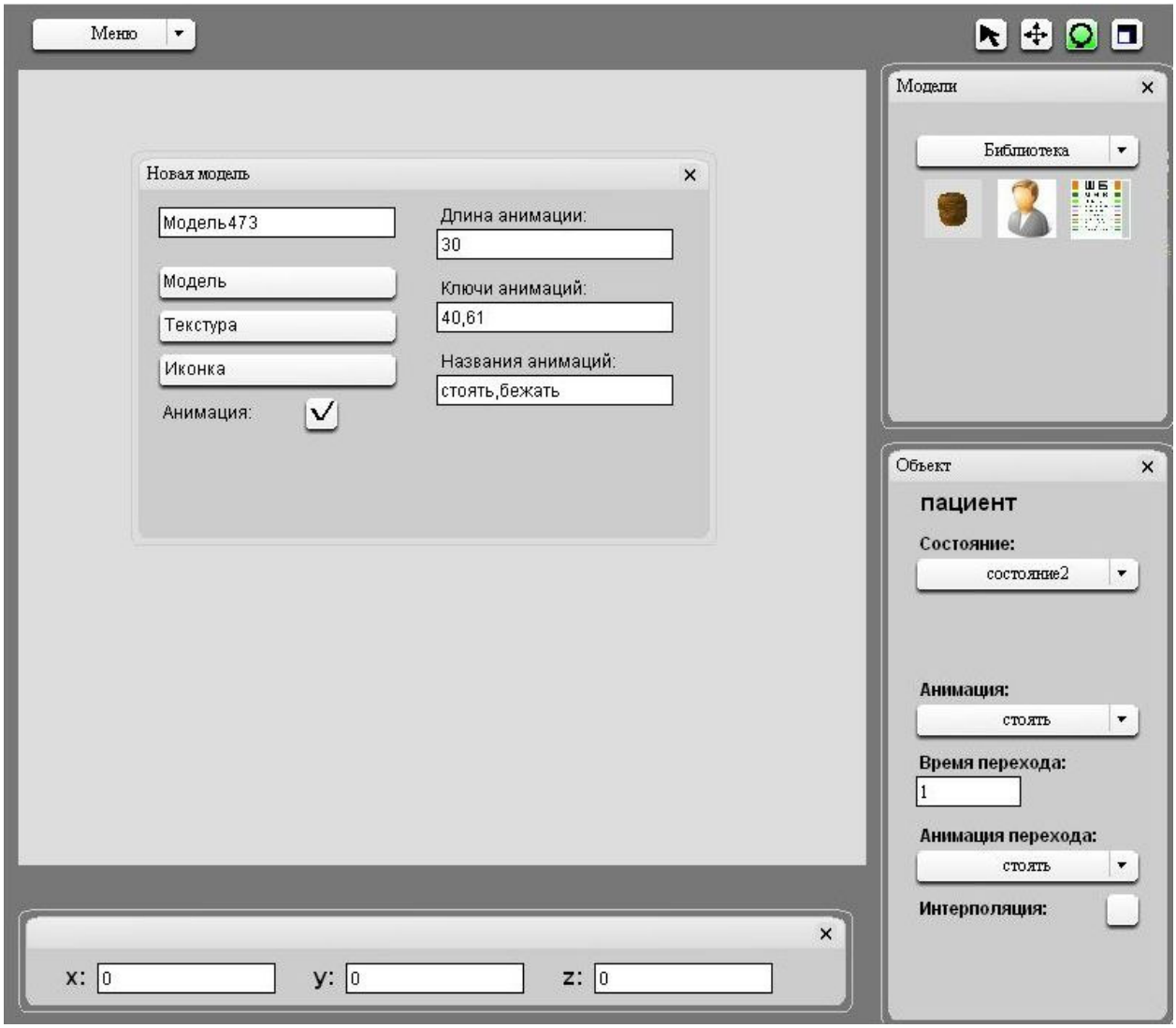


Рис. 5.1.4. Добавление новой модели

Размещение объекта в пространстве сцены (определение его координат) производится с помощью инструмента "координаты". При этом к объекту привязываются координатные оси: красная - ось X, зеленая – ось Y, и синяя – о ось Z. Перемещая эти стрелки, дизайнер перемещает объект в виртуальном пространстве сцены. Полученные координаты объекта автоматически отображаются в нижней панели интерфейса редактора, в которой может быть также осуществлен и ручной ввод требуемых точных координат (см. рис. 5.1.5).

Поворот объекта в пространстве (определение углов его поворота вокруг трех осей) производится с помощью инструмента "повороты" (кнопка с иконкой окружности).

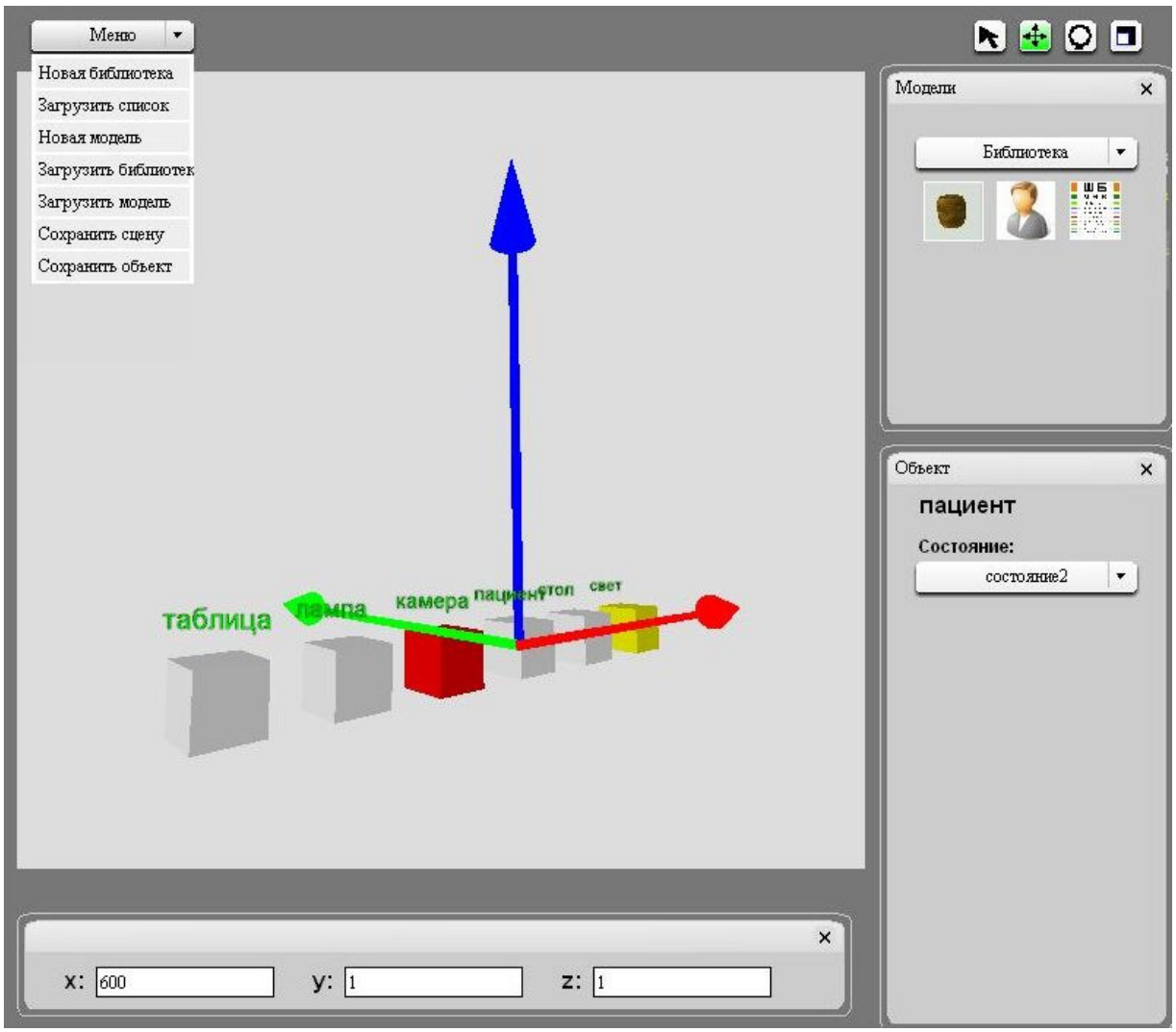


Рис. 5.1.5. Графический редактор

Масштабирование объекта в пространстве (определение коэффициентов масштабирования вдоль трех осей) производится с помощью инструмента "масштабирование". Во всех описанных выше инструментах дизайнер либо средствами визуального программирования, либо через соответствующие поля ввода определяет указанные выше атрибуты презентационного уровня.

Некоторые объекты сцены могут находиться в разных **состояниях**, которые определил для него эксперт предметной области. Дизайнер может определить уникальное представление объекта на сцене для каждого заданного состояния.

Все описанные экспертом состояния отображаются в редакторе в виде выпадающего списка в окне параметров выделенного объекта (см. Рис. 5.1.6).

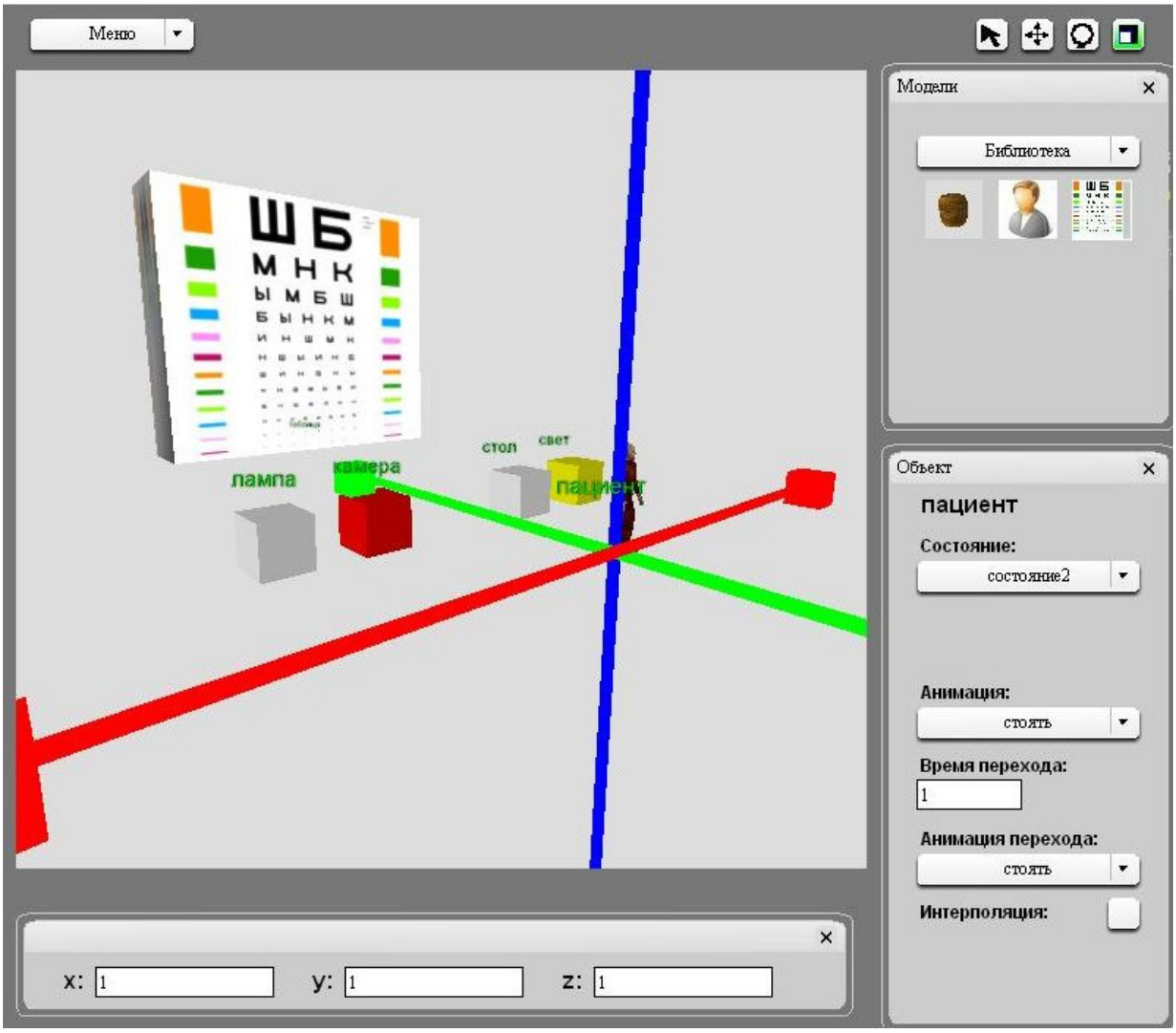


Рис. 5.1.6. Состояния объекта

Все действия, производимые дизайнером с объектом сцены, относятся к выбранному состоянию объекта (если у объекта есть состояния). Таким образом, для определения нужных презентационных свойств объекта в некотором состоянии, достаточно выбрать это состояние из списка и работать с объектом как обычно: все операции перемещения, поворотов, масштабирования записываются только для выбранного состояния объекта.

При выборе другого состояния объекта автоматически выполняется его презентационное преобразование в соответствии с данными, записанными для этого (выбранного) состояния

По умолчанию переход из одного состояния в другое осуществляется моментально (дискретно), т.е. новые презентационные параметры объекта сразу заменяют старые и отображение объекта мгновенно меняется. Однако, для

перехода из одного состояния в другое может быть определена анимация непрерывного изменения презентационных параметров в течение некоторого времени. Для этого в блоке параметров объекта для выбранного состояния вводится необходимое значение времени, в течение которого объект будет "переходить" в данное состояние из предыдущего (см. Рис. 5.1.7).

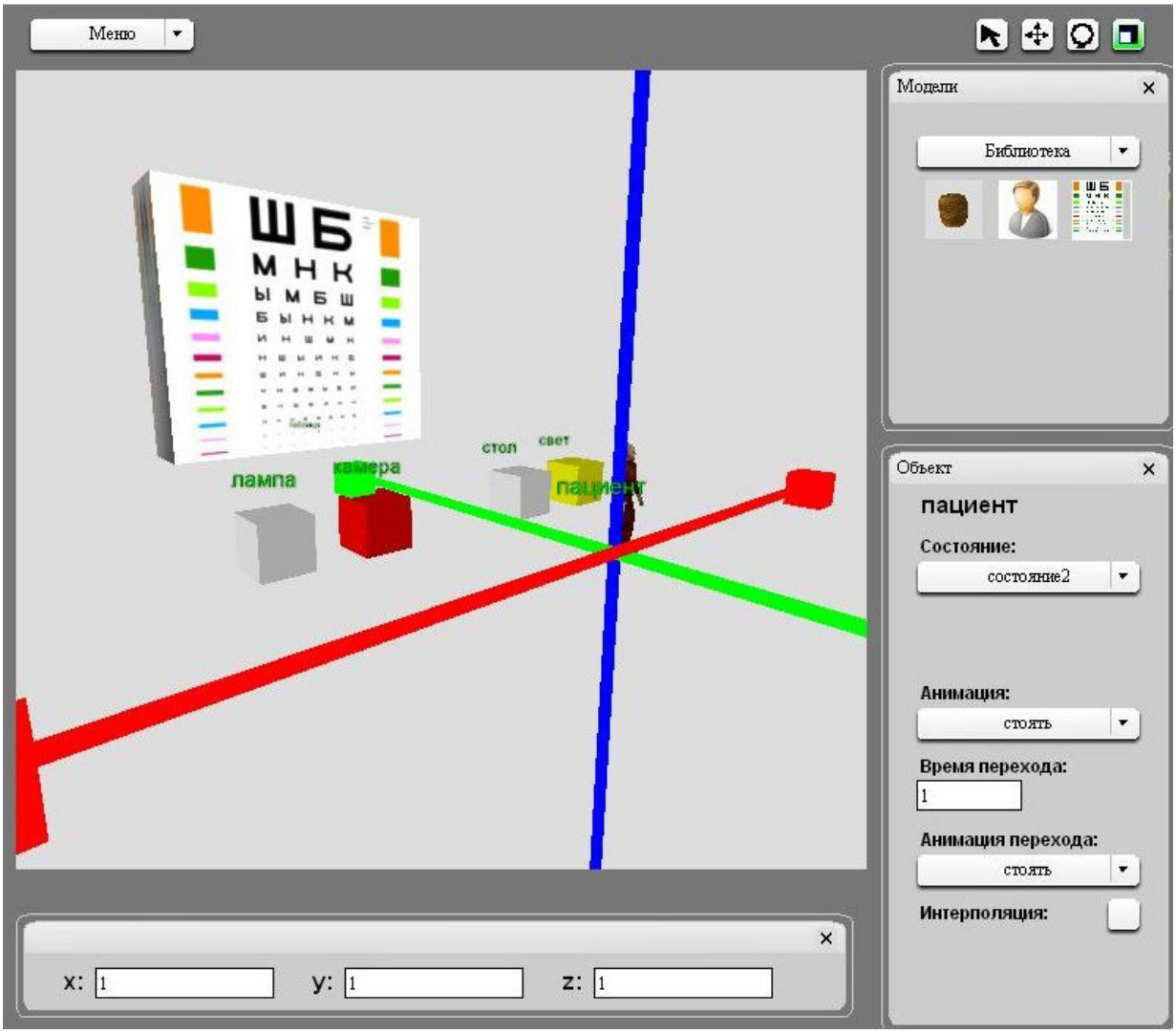


Рис. 5.1.7. Определение непрерывного анимированного перехода

По своей структуре трехмерные модели могут быть статичные и анимационные (со скелетной анимацией). Анимационные модели имеют внутреннюю (заданную для мэша) анимацию и могут быть анимированы не только при переходах из одного состояния в другое, но также и в обычном положении.

Для загрузки анимационной трехмерной модели в библиотеку необходимо указать не только имя, файлы мэша, текстуры и иконки, но также и

дополнительные параметры анимаций модели: длину анимационной последовательности, ключевые кадры анимаций, и отображаемые в редакторе имена для соответствующих ключевых анимационных последовательностей (см. Рис. 5.1.8).

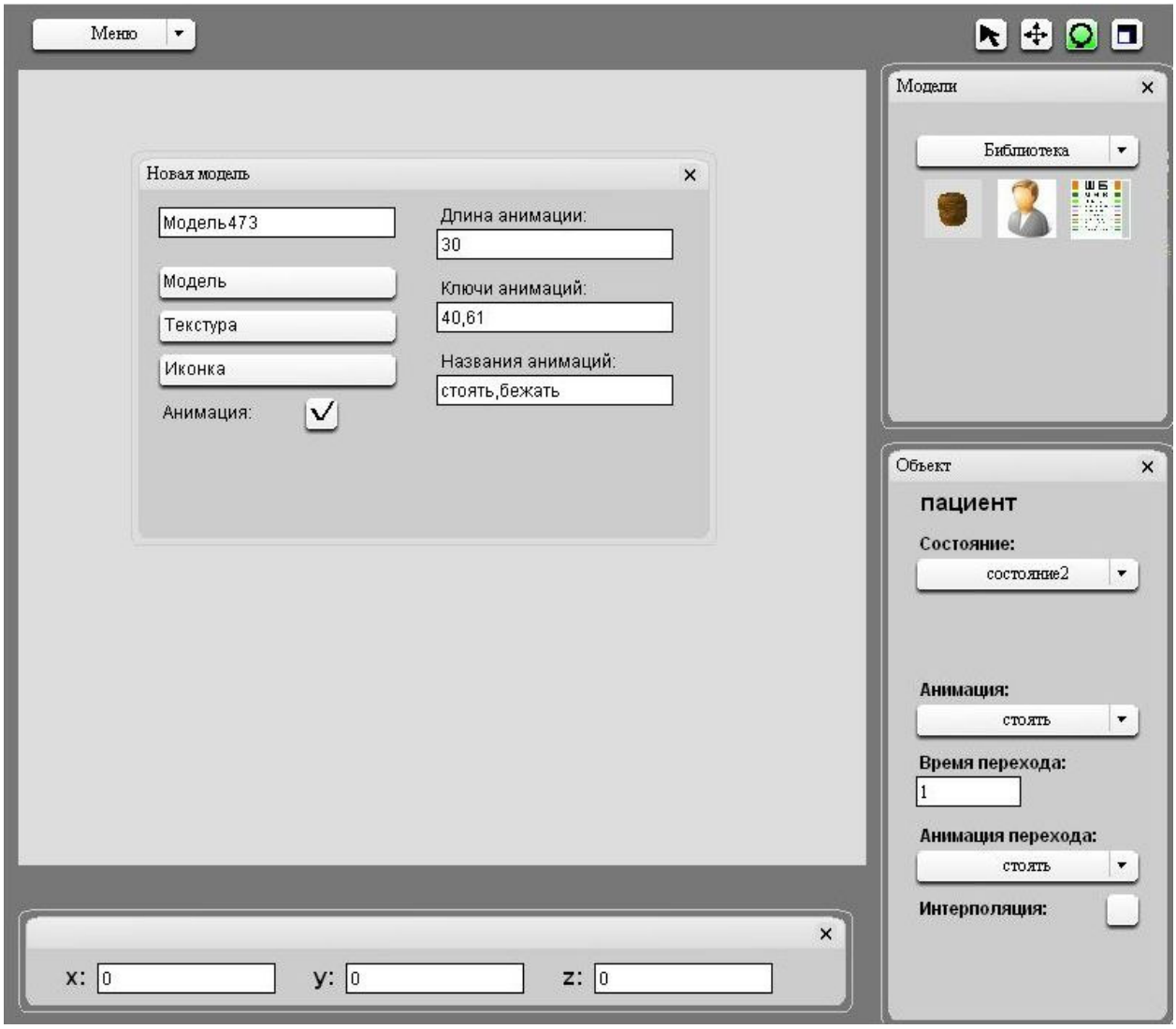


Рис. 5.1.8. Загрузка анимационной модели

Объект сцены, для которого выбирается анимационная 3d-модель, как правило, имеет множество состояний. Для такого объекта для каждого состояния можно определить не только время непрерывного перехода как для неанимационных моделей, а также и необходимые анимации для процесса перехода в состояние и для самого состояния (т.е. каким образом должен быть анимирован объект, когда примет данное состояние)

Определение элементов таблицы

Объект "таблица" отличается от остальных объектов тем, что содержит в своем составе множество подобных элементов.

Эксперт предметной области определил множество необходимых элементов таблицы. Дизайнер в редакторе должен определить презентационное представление этих элементов для выбранного табличного объекта. Презентационное представление каждого элемента таблицы – это прямоугольная область, наложенная на одну из плоских граней объекта (см. Рис. 5.1.9).

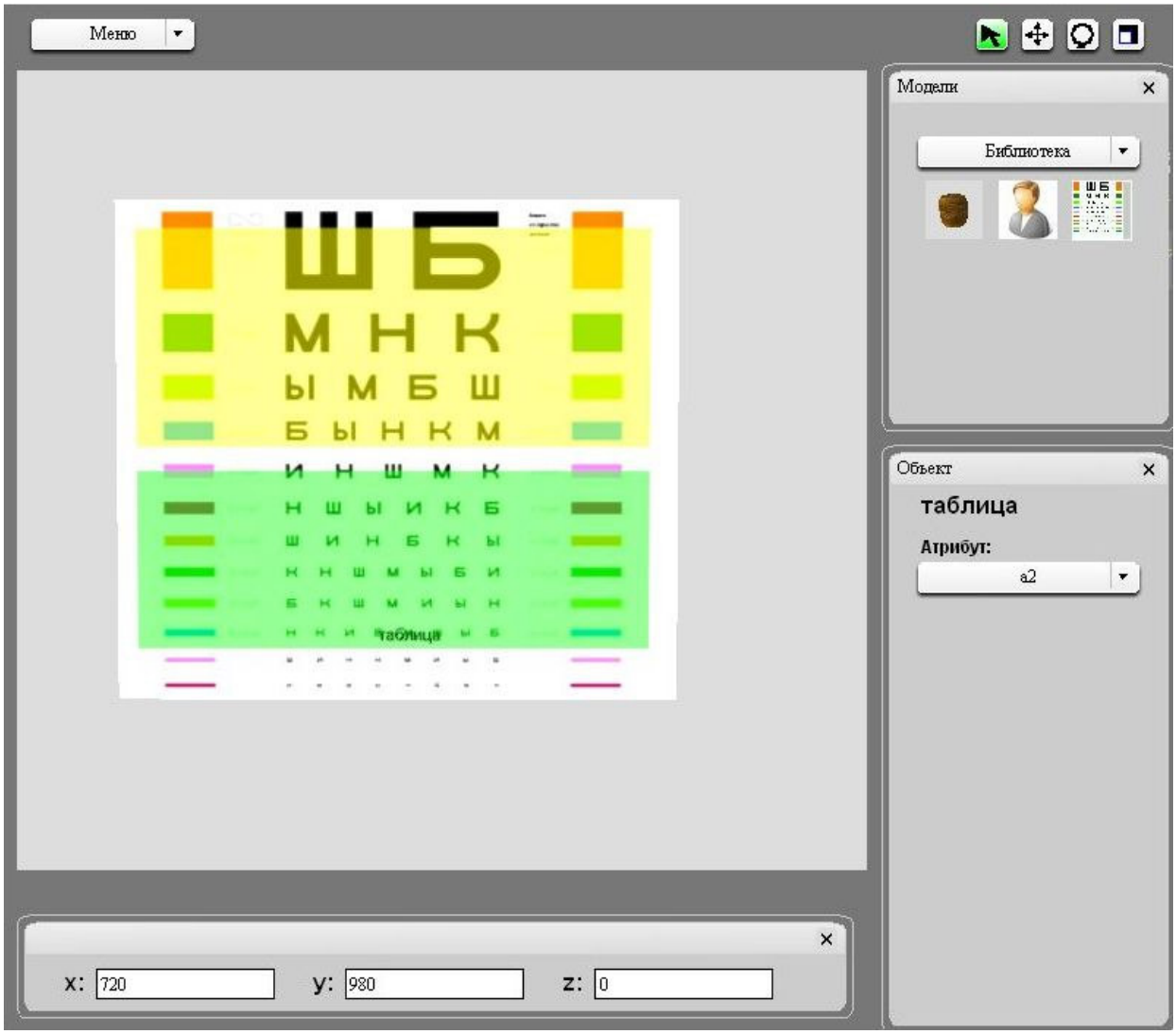


Рис. 5.1.9. Элементы объекта "таблица"

Все созданные экспертом элементы таблицы отображаются в виде выпадающего списка в блоке параметров объекта. Дизайнер выбирает из

данного списка нужный элемент и с помощью мышки указывает на объекте координаты двух углов выбранного элемента: верхний левый угол, правый нижний. Таким образом, для данного элемента таблицы определяется его прямоугольная область. Область текущего элемента окрашивается в зеленый цвет. Области, определенные для других элементов таблицы окрашиваются в желтый цвет, чтобы видеть, какое пространство объекта уже заполнено.

Технология разработки агентов

В описании декларативной модели ПВС могут понадобиться параметры, которые должны вычисляться специфическим (нестандартным) способом. В этом случае такой параметр помечается как "параметр обработки агентом", для него в модели указывается имя, описание и возвращаемое значение, В общем случае программная реализация специфической обработки параметра является независимой задачей и может быть выполнена отдельным внешним независимым модулем. Такие модули в системе формируются как агенты (платформа IACPaaS реализована как мультиагентная).

Написание агента для программиста, по сути, представляет собой написание некоторой независимой функции с известными входными и выходными данными. Входными данными для агента является динамическая версия декларативной модели виртуальной среды и сообщение, переданное от веб-клиента. Требуемые параметры среды и параметры сообщения веб-клиента считаются известными программисту (он должен их получить из описания вычисляемого параметра).

Проверка целостности модели

Проверка целостности разработанной декларативной модели профессиональной виртуальной среды состоит из двух уровней. Первый уровень заключается в том, что декларативная модель строится на основе онтологии, содержащей набор ограничений целостности, который проверяется автоматически при создании декларативной модели, используя структурный редактор. Таким образом, первый уровень проверки целостности обеспечивает синтаксическую проверку целостности модели. Второй уровень проверки

целостности осуществляется на программном уровне в графическом редакторе и интерпретаторе, в каждом из которых проверяется наличие всех необходимых на данный момент сущностей декларативной модели для воспроизведения, а также их корректность, т.е. наличие необходимых ресурсов, ссылок, отсутствие перекрестных ссылок и т.п.. Таким образом, второй уровень проверки целостности обеспечивает частичную семантическую проверку целостности декларативной модели.

5.2. Использование профессиональных виртуальных сред

Интерпретирующая система (интерпретатор) предназначается для запуска и работы созданных профессиональных виртуальных сред. Каждое разработанное прикладное приложение становится отдельным сервисом облачной платформы. Поэтому для его использования необходимо зайти на платформу, выбрать необходимую предметную область, найти сервис и запустить его в браузере. Для использования разработанных виртуальных сред пользователи получают необходимые полномочия для требуемого приложения аналогично полномочиям для других сервисов: выбирают нужную им предметную область (в которой находится требуемая ПВС), и делают заявку на использование сервиса. После получения необходимых полномочий пользователи получают удаленный доступ к ПВС в соответствии с моделью SaaS.

Пользователи ПВС могут выполнять следующие задачи:

1) Определять параметры виртуальной среды (если таковые имеются) для разнообразия выполняемых в ней действий.

2) Исследовать виртуальный интерактивный мир запущенной ПВС.

3) Выполнить необходимые действия в виртуальной среде и записать их на сервере платформы для последующего просмотра.

4) Просматривать сохраненные ролики по данной виртуальной среде (как свои, так и чужие ролики).

5.3. Экспериментальное исследование программного комплекса

Экспериментальное исследование программного комплекса производилось в ходе разработки представленных ниже виртуальных сред: тренажеров с виртуальной реальностью для офтальмологии, виртуальной химической лаборатории, демонстрационного проекта городского района.

5.3.1. Компьютерный обучающий тренажер с виртуальной реальностью для офтальмологии

Компьютерные обучающие системы являются высоко востребованными в рамках реализации Федеральных государственных образовательных стандартов третьего поколения [61, 92]. Разработанный компьютерный обучающий тренажер [31, 36-37] включает обучающие задания по классическим методам исследования в офтальмологии: определение остроты зрения по таблицам, определение остроты зрения по оптотипам Б.Л. Поляка, определение клинической рефракции объективным методом (скиаскопия), исследование поля зрения методом кампиметрии. Приведены примеры декларативных моделей разработанных тренажеров.

В каждом обучающем задании студенту необходимо продемонстрировать знание технологии проведения некоторого исследования и по поведению виртуального пациента определить значение некоторого признака, определяемого обучающим заданием. Значение признака автоматически генерируется по описанию множества его возможных значний в модели ПВС.

В качестве примера в диссертационной работе приведено описание метода исследования определения остроты зрения по таблицам.

При проведении исследования по определению остроты зрения студент должен в правильном, строго определенном порядке выполнить ряд действий (эти требования определяются процедурой проведения исследования), например, включить аппарат рота, провести исследования сначала правого глаза, затем левого, при этом методика опроса также имеет свою последовательность. В зависимости от ответов пациента (ответы автоматически генерируются по описанию декларативной модели), студент должен либо

продолжить проведение исследования, либо выдать значение признака – остроту зрения. Сгенерированные ответы пациентов могут, в зависимости от остроты зрения, быть следующих типов: все буквы названы правильно, несколько букв на строке названы с ошибками (количество неправильно названных букв также определяется остротой зрения), либо виртуальный пациент генерирует ответ «не вижу». По окончании исследования студент может сравнить свой результат с правильным, а также получить объяснение его действиям (если они были выполнены неправильно либо в неправильном порядке).

На рис. 5.3.1 приведена обобщенная схема логического представления модели, сформированная экспертами предметной области. Подробная декларативная модель представлена в Приложении 3.

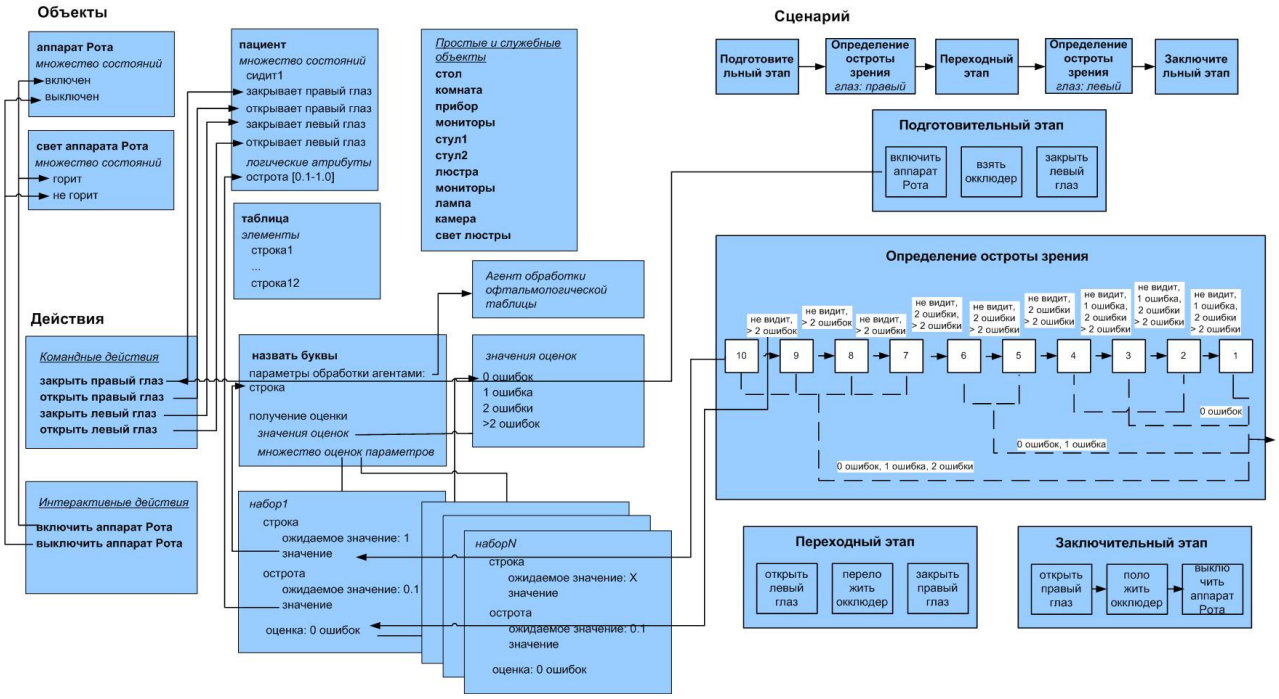


Рис. 5.3.1. Обобщенная схема логического представления модели

На рис. 5.3.2 представлен процесс разработки графического отображения разработанного тренажера.

Для описанного метода обучения определения остроты зрения по таблицам Сивцева экспертом предметной области (врачом) были сформированы декларативные модели реализованных тренажеров, в том числе,

описано 10 объектов, 10 действий и сценарий. Дизайнером в сторонних редакторах (3ds max, Photoshop) были подготовлены графические материалы для тренажеров: 10 трехмерных моделей, 10 текстур; и сформирована итоговая виртуальная сцена. Программистом был написан агент обработки таблицы проверки остроты зрения.

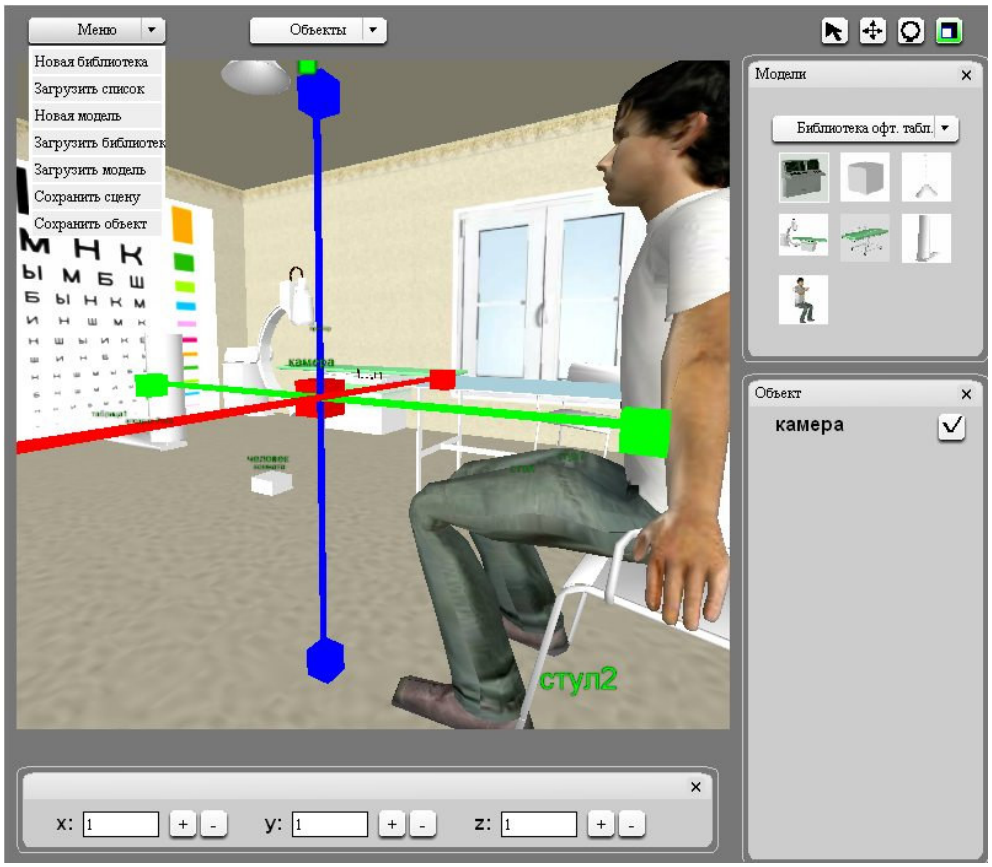


Рис. 5.3.2. Редактирование виртуальной среды тренажера для обучения определению остроты зрения по таблицам

В качестве еще одного примера приведено описание тренажера по обучению исследования поля зрения методом кампиметрии.

При проведении исследования по определению поля зрения методом кампиметрии студент должен аналогично предыдущему примеру в правильном, строго определенном порядке выполнить ряд действий: надеть повязку пациенту на один глаз, попросить пациента поставить голову на упор, провести исследования сначала правого глаза, затем левого, при этом методика опроса также имеет свою последовательность. В зависимости от ответов пациента (ответы автоматически генерируются по описанию декларативной модели в зависимости от неизвестной величины – длины очередного видимого

меридиана), студент должен либо продолжить проведение исследования, либо выдать значение признака – поле зрения. Сгенерированные ответы пациентов могут, в зависимости от поля зрения, быть следующих типов: «вижу» или «не вижу». По окончании исследования студент может сравнить свой результат с правильным, а также получить объяснение его действиям (если они были выполнены неправильно либо в неправильном порядке).

На рис. 5.3.3 приведена обобщенная схема логического представления модели тренажера исследования поля зрения методом кампиметрии, сформированная экспертами предметной области. Подробная декларативная модель представлена в Приложении 3.

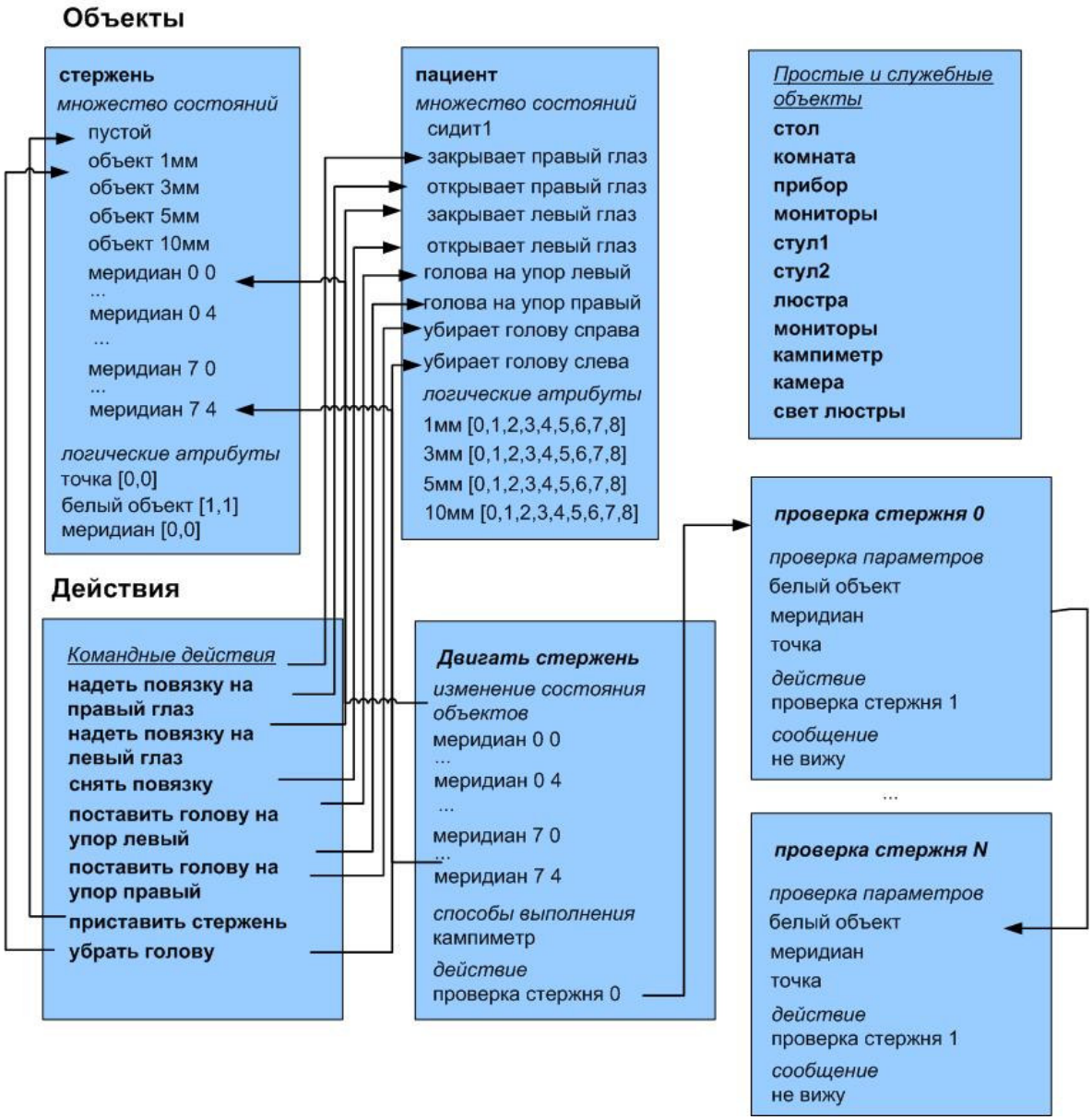


Рис. 5.3.3. Обобщенная схема логического представления модели

На рис. 5.3.4 представлен процесс разработки графического отображения разработанного тренажера по исследованию поля зрения методом кампиметрии.

Применение инструментального средства разработки ПВС позволило значительно сократить трудоемкость разработки и упростить последующее сопровождение, при этом вовлечь экспертов предметной области (врачей) в процесс разработки.

Это удалось за счет следующих возможностей инструментального комплекса:

- Разделения разработки ПВС между различными специалистами: экспертом (врачом), дизайнером и программистом.
- Повторного использования одних и тех же декларативных элементов различных тренажеров (например, использование одних и тех же предметов интерьера и медицинских инструментов).
- Повторного использования ресурсов библиотек (трехмерных моделей, текстур).
- Доступа к разрабатываемой ПВС через Интернет, так как различные специалисты смогли принять участие в разработке единой программы, находящейся в едином месте (на платформе) из разных мест.

Применение разработанных тренажеров позволит упростить обучение студентов медицинских вузов, обучающихся по специальности "офтальмология". В настоящее время тренажер внедрен на кафедре офтальмологии и оториноларингологии для обучения студентов лечебного факультета по курсу "офтальмология" и дистанционного обучения врачей. Акт внедрения в Приложении 4.

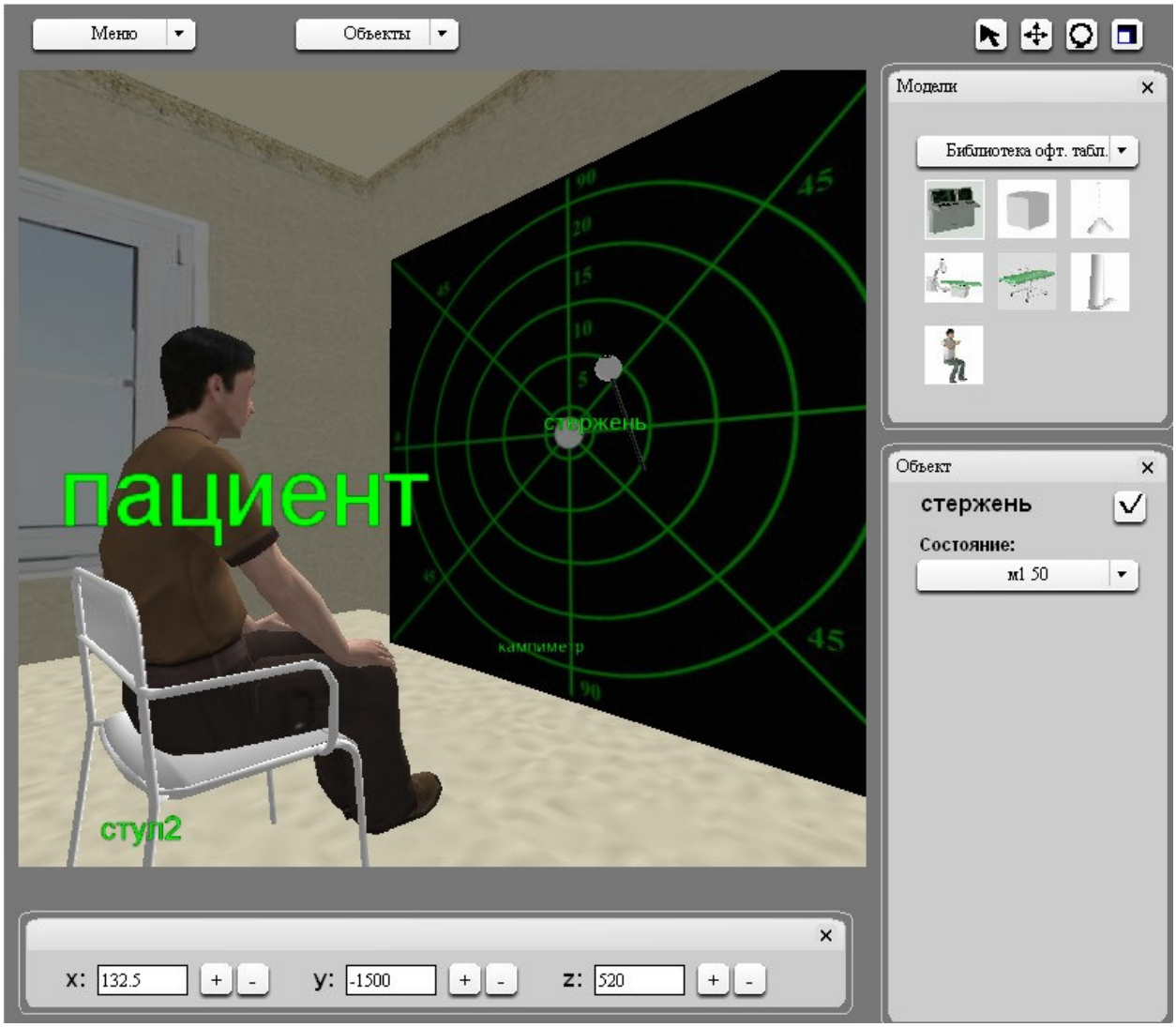


Рис. 5.3.4. Редактирование виртуальной среды тренажера для обучения исследованию поля зрения методом кампиметрии

5.3.2. Виртуальная химическая лаборатория

Виртуальная химическая лаборатория предназначена для безопасного проведения химических опытов и экспериментов из школьной практики.

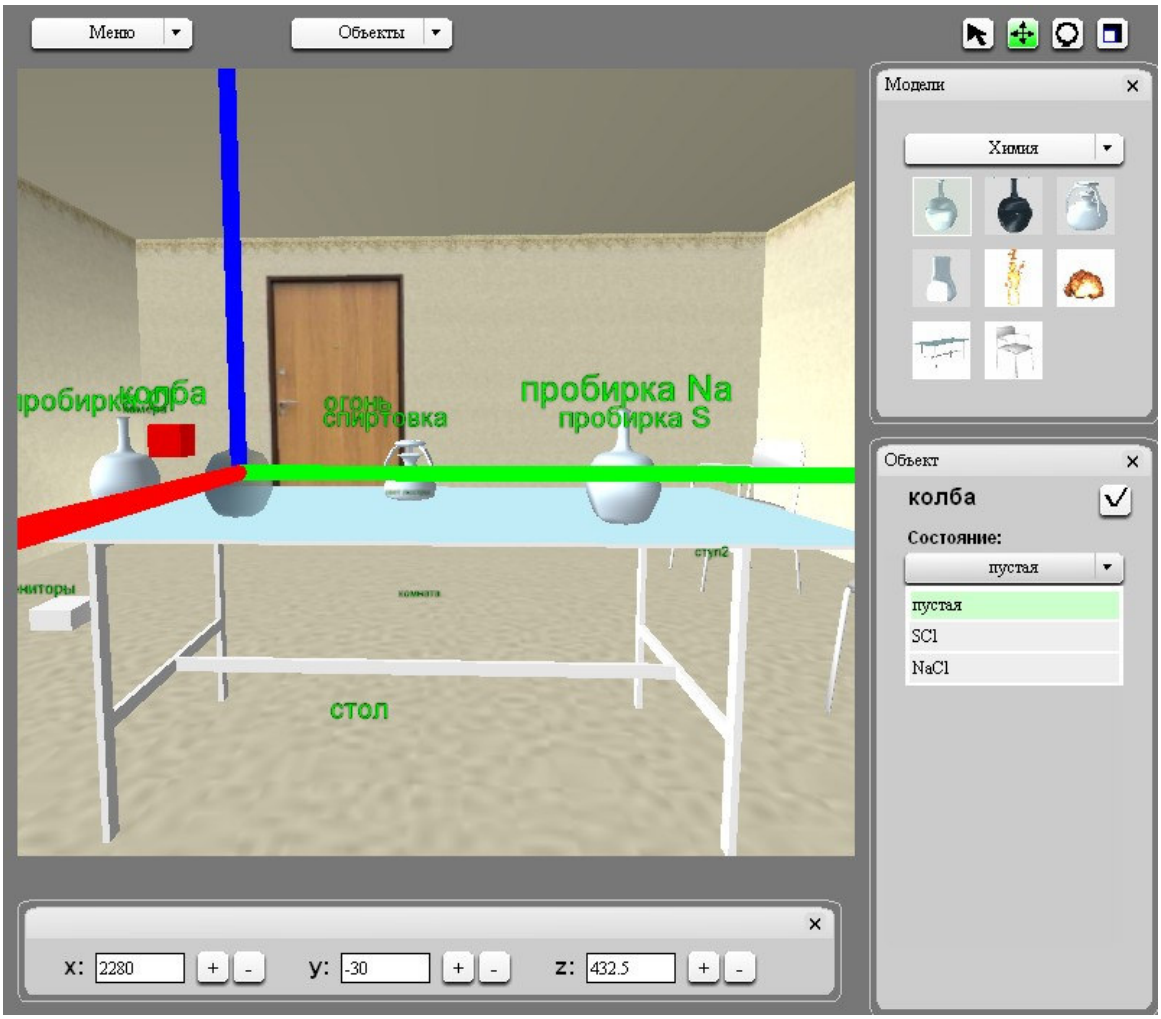


Рис. 5.3.5. Редактирование виртуальной химической лаборатории

На рис. 5.3.5 представлен пример редактирования виртуальной среды для химической лаборатории. Подробное описание данной виртуальной среды см. в Приложении 3.

5.3.3. Демонстрационный проект городского района

Демонстрационный проект городского района предназначен для визуальной интерактивной демонстрации объектов запланированного под строительство нового жилого комплекса и позволяет показать клиентам все преимущества покупки недвижимости в данном районе.

5.4. Выводы

В результате выполнения пятой задачи диссертационной работы разработана технология создания профессиональных виртуальных сред с помощью реализованного инструментального сервиса для различных

специалистов: экспертов предметной области, дизайнеров, программистов. Совместно с экспертами предметной области разработаны следующие облачные сервисы: компьютерный обучающий тренажер по классическим методам офтальмологии, виртуальная химическая лаборатория, виртуальная демонстрация городского района.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

1. На основе анализа современных проблем при разработке профессиональных виртуальных сред предложены новые принципы их разработки и функционирования. В соответствии с этими принципами проектирование, реализация и сопровождение профессиональной виртуальной среды (ПВС) на языке программирования заменяется проектированием и сопровождением декларативной модели ПВС с последующей ее интерпретацией; в процесс разработки включаются дизайнеры и эксперты предметной области, которые формируют модель ПВС по ее онтологии; средство разработки модели ПВС и сама виртуальная среда предоставляются разработчикам и пользователям как облачные сервисы.

2. Разработана онтология профессиональной виртуальной среды, включающая описание объектов, действий, сценариев. В онтологии выделены логический и презентационный уровни, позволяющие не только включить в разработку дизайнеров и экспертов предметной области, но также и разделить работу между ними.

3. Разработаны и классифицированы различные типы моделей профессиональных виртуальных сред: неинтерактивные статичные, предназначенные для демонстрации каких-либо предметов, статичных сцен; неинтерактивные анимационные, в которых объекты сцены имеют анимацию; интерактивные виртуальные среды без контроля – среды, позволяющие пользователю взаимодействовать с объектами сцены, получая от них обратную реакцию; интерактивные с контролем – среды, в которых анализируются действия пользователя и выдается объяснение полученных результатов.

4. Разработаны методы интерпретации модели ПВС, которые включают в себя создание виртуальной сцены, обработку событий, изменение объектов сцены. Определяется механизм взаимодействия между клиентской и серверной частью виртуальной интерактивной среды, форматы сообщений между различными модулями, механизм хранения данных.

5. Разработаны методы реализации инструментального комплекса. Реализован интерпретатор, клиентский и серверный модули инструментария. Разработан 3d-редактор для создания трехмерных сцен виртуальных сред. Реализованный инструментальный комплекс работает как облачный сервис на базе платформы IACPaaS.

6. Разработана технология создания профессиональных виртуальных сред с помощью реализованного инструментального сервиса для различных специалистов: экспертов предметной области, дизайнеров, программистов. Совместно с экспертами предметной области разработаны следующие облачные сервисы: компьютерный обучающий тренажер по классическим методам офтальмологии, виртуальная химическая лаборатория, виртуальная демонстрация городского района.

ЛИТЕРАТУРА

1. Адеев М. Учебник по VRML 97. [Электронный ресурс]. URL: http://citforum.ru/internet/vrml/ (дата обращения: 10.06.2012)
2. Артемьева И.Л., Клещев А.С. Математические модели онтологий предметных областей. Ч. 1. Существующие подходы к определению понятия «онтология» // НТИ. Сер. 2. 2001. – №2. – С.20–26.
3. Ахремчик О.JL, Бодрин А.В. Фреймовый подход к созданию моделей систем автоматизации // Сборник научных трудов "Компьютерные технологии в управлении и диагностике" . –ТГТУ, Тверь, 2004. – С. 112-115.
4. Ахремчик О. Л., Филатова Н. Н., Вавилова Н. Н., Тулова С.А. Тренажеры для обучения разработке систем автоматизации, реализованные в WWW // Proceedings EEEE International Conference on Advanced Learning Technologies. – Kazan. – 2002. – pp. 288-292.
5. Баяковский Ю.М., Игнатенко А.В., Фролов А.И. Графическая библиотека OpenGL. Учебно-методическое пособие, Москва: Издательский отдел факультета Вычислительной Математики и Кибернетики МГУ им. Ломоносова. 2003. – 132 с.
6. Беневоленский С.Б., Марченко А.Л. Использование виртуальных тренажеров в процессе изучения электротехнических дисциплин // Педагогическая информатика. – 2009. – №3. – С. 24-30.
7. Бодрин, А.В., Разработка моделей и алгоритмов синтеза схем для тренажерного комплекса, обучающего проектированию систем автоматизации // диссертация на соискание уч. степени к.т.н., 05.13.12, http://www.dissercat.com/content/razrabotka-modelei-i-algoritmov-sinteza-skhem-dlya-trenazhernogo-kompleksa-obuchayushchego-p
8. Борисов В. Г., Данилова С. К., Чинакал В. О. Исследования по созданию перспективных систем управления морскими подвижными

объектами и разработке тренажерных систем // Проблемы управления. – 2009.– №3.1. – С. 103–106

9. Борисов В.Г., Данилова С.К., Чинакал В.О. Применение средств виртуальной реальности при создании комплексных тренажеров и систем управления // Труды международной конференции “Системы проектирования, технологической подготовки производства и управления этапами жизненного цикла промышленного продукта (CAD/CAM/PDM – 2009)”. – М.: ИПУ РАН. – 2009
10. Борисов В. Г., Данилова С. К., Чинакал В. О. Создание и применение компьютерной технологии повышения безопасности управления морскими подвижными объектами // Проблемы управления. – 2007. – №4.– С. 79–84
11. Вавилова Н.И., Ахремчик О.Л. Два уровня моделей в мультимедиа тренажерах // Сборник трудов международной научной конференции "Математические методы в интеллектуальных системах", ММИИСТ-2002. – Смоленск . – 2002 – С. 20.
12. Вавилова Н.И. Виртуальные тренажеры // Сборник трудов международной конференции ММТТ-2000. – Санкт-Петербург 2000. – С. – 10-11.
13. Вавилова Н.И. Модели и алгоритмы автоматизированного проектирования макетов сцен мультимедиа тренажеров // диссертация на соискание уч. степени к.т.н., 05.13.12, http://www.dissercat.com/content/modeli-i-algoritmy-avtomatizirovannogo-proektirovaniya-maketov-stsen-multimedia-trenazherov
14. Вавилова Н.И. Мультимедиа тренажеры // Материалы конференции "Развитие новых технологий в системе открытого образования России". – Тверь. – 2001. – С. 4-5.
15. Вавилова Н.И. Проектирование моделей тренажера на основе онтологического подхода // Сборник материалов всероссийской

заочной конференции "Перспективы развития волжского региона", Тверь 2002. – №.4. – С. 142-146.

16. Вавилова Н.И., Филатова Н.Н. Принципы создания мультимедиа тренажеров // Сборник научных трудов ТГТУ "Проектирование технических и медико-биологических систем", Тверь. – 2000. – С. 4-7.
17. Васюгова С.А., Варламов О.О. Информационное общество: исследование перспектив и проблем интеграции человека с компьютером. Технологическая сингулярность как новый этап обучения в образовании // Материалы международной конференции "Информация и образование: границы коммуникаций", INFO'11. – Горно-Алтайск : РИО ГАГУ. – 2011. – С. 30-33
18. Виртуальная регата [Электронный ресурс] — URL: http://www.eligovision.ru/casestudy/14/
19. Виртуальный симулятор ЛапСим [Электронный ресурс] — URL: http://rusalfamed.com/index.php/ru/another-notes/21-simulyasionnoe-oborudovanie/113-lapsim
20. Военные и геймеры обменялись технологиями [Электронный ресурс] - — URL: http://rnd.cnews.ru/news/line/index_science.shtml?2013/02/12/518841
21. Гаммер М.Д. и др. Компьютерные имитационные тренажеры в открытом профессиональном образовании // Открытое образование. – №5. –2009. – С. 48-52
22. Гаммер М.Д. и др. Опыт проектирования и использования компьютерных имитационных тренажеров в ТЮМГНГУ, [Электронный ресурс] – URL: http://lib.znate.ru/docs/index-87944.html?page=6
23. Гаммер М.Д. Применение компьютерных имитационных тренажеров и систем виртуальной реальности в учебном процессе [Электронный ресурс] – URL: http://www.really.ru/articles/events/123--vr-.html

24. Гаммер, М. Д. Разработка системы автоматизированного проектирования компьютерных имитационных тренажеров: автореферат диссертации на соискание ученой степени канд. техн. наук :05.13.12 / М. Д. Гаммер. - Тюмень, 2007. - 19 с.
25. Грибова В.В., Клещев А.С., Крылов Д.А., Москаленко Ф.М., Тимченко В.А., Шалфеева Е.А. Агентный подход к разработке интеллектуальных Интернет-сервисов // Труды конгресса по интеллектуальным системам и информационным технологиям "IS&IT'12" (2-9 сентября 2012 г., Россия, Дивноморское). - М.: Физматлит. –2012. Т.1. - С. 218-223.
26. Грибова В.В., Клещев А.С., Крылов Д.А., Москаленко Ф.М., Смагин С.В., Тимченко В.А., Тютюнник М.Б., Шалфеева Е.А. Облачная платформа для разработки и управления интеллектуальными системами // Материалы международной научно-технической конференции "Открытые семантические технологии проектирования интеллектуальных систем". – Минск: БГУИР. – 2011. – С. 5-14.
27. Грибова В.В., Клещев А.С., Крылов Д.А., Москаленко Ф.М., Смагин С.В., Тимченко В.А., Тютюнник М.Б., Шалфеева Е.А. Проект IACPaaS. Комплекс для интеллектуальных систем на основе облачных вычислений // Искусственный интеллект и принятие решений. – 2011. – № 1. – С.27-35.
28. Грибова В.В., Клещев А.С., Крылов Д.А., Москаленко Ф.М., Смагин С.В., Тимченко В.А., Тютюнник М.Б., Шалфеева Е.А. Развиваемый интернет-комплекс для разработки, управления и функционирования интеллектуальных систем // Труды Пятой международной конференции «Параллельные вычисления и задачи управления» (PACO’2010) [Электронный ресурс]. Москва, – 2010. – С. 1415-1421.
29. Грибова В.В., Клещев А.С. Онтологическая парадигма программирования // II международная научно-техническая конференция "Открытые семантические технологии проектирования

интеллектуальных систем" (OSTIS-2012). – Минск: БГУИР. – 2012. – С. 213-220.

30. Грибова В.В., Осипенков Г.Н., Сова С.А. Концепция разработки диагностических компьютерных тренажеров на основе знаний //International Book Series "Human Aspectsof Artifical Intelligence". N12. Papers are selected from Proc. of the Intern. Conf. of the Join International Events of Informatics “ITA 2009” (e.TECH-2009), Varna, Bulgaria, 2009. P. 27-33. – Suppl. to the International Journal “Information Technologies & Knowledge”. – Vol.32. Sofia, Bulgaria (Inst. of Information Theories and Applications FOI ITHEA) . – 2009.
31. Грибова В.В., Петряева М.В., Федорищев Л.А. Разработка виртуального мира медицинского компьютерного обучающего тренажера // Дистанционное и виртуальное обучение. – 2011. – № 9. – С. 56-66.
32. Грибова В.В., Петряева М.В., Федорищев Л.А., Черняховская М.Ю. Модель виртуального мира мультимедиа тренажера для медицинского образования // International Book Series "Information Science and Computing". - Applicable Information Models (MeL-2011). –№ 22. – 2011. – Sofia, – Bulgaria. – Pp. 140-148.
33. Грибова В.В., Петряева М.В., Федорищев Л.А., Черняховская М.Ю. Модель объектов виртуального мира для диагностических медицинских компьютерных тренажеров. - Владивосток: ИАПУ ДВО РАН, 2010. - 24 с.
34. Грибова В.В., Петряева М.В., Федорищев Л.А., Черняховская М.Ю. Структура формального представления объектов в компьютерных диагностических тренажерах // Сборник статей Второй международной научно-практической конференции “Высокие технологии, фундаментальные и прикладные исследования в физиологии и медицине”. 26–28 10.2011, Санкт-Петербург, Россия /

под ред. А.П. Кудинова, Б.В. Крылова. – СПб: Изд-во Политехн. ун-та, 2011. Т. 2. – С. 103-105.

35. Грибова В.В., Петряева М.В., Федорищев Л.А., Черняховская М.Ю. Формализация методов исследования в офтальмологии для компьютерных диагностических тренажеров // Сборник статей Третьей международной научно-практической конференции «Высокие технологии, фундаментальные и прикладные исследования в физиологии и медицине». – СПб.: Изд-во Политехн. ун-та, 2012. – Т.2. – С.191-195.
36. Грибова В.В., Петряева М.В., Федорищев Л.А., Черняховская М.Ю. Формальное представление методов исследования в офтальмологии для медицинских обучающих систем Ч.1. - Владивосток: ИАПУ ДВО РАН, 2012.- 28 с.
37. Грибова В.В., Петряева М.В., Федорищев Л.А., Черняховская М.Ю. Формальное представление методов исследования в офтальмологии для медицинских обучающих систем Ч.2. - Владивосток: ИАПУ ДВО РАН, 2012.- 36 с.
38. Грибова В.В., Федорищев Л.А. Виртуальная реальность в образовании: система разработки интернет-проектов // Материалы международной научно-практической конференции «Новые информационные технологии в образовании». – Екатеринбург: ФГАОУ ВПО «Рос. гос. проф.-пед. ун-т». – 2012. – С.116-118.
39. Грибова В.В., Федорищев Л.А. Инструментальный сервис для создания виртуальных интерактивных облачных сред // International Journal «Information Technologies & Knowledge». – 2013. – Vol .7, №3. – Pp. 277-281.
40. Грибова В.В., Федорищев Л.А. Интернет-комплекс для создания обучающих систем с виртуальной реальностью // Дистанционное и Виртуальное Обучение. – 2012. – № 7. – С. 4-12.

41. Грибова В.В., Федорищев Л.А. Интерпретатор проекта виртуальной среды для интерактивных компьютерных систем в Интернете // материалы XIX международной заочной научно-практической конференции «Технические науки — от теории к практике». – Новосибирск: Изд. “СибАК». – 2013. – С. 7-14.
42. Грибова В.В., Федорищев Л.А. Обучающие виртуальные системы и средства их создания // Вестник информационных и компьютерных технологий. – 2012. – №3. – С. 48-51.
43. Грибова В.В., Федорищев Л.А. Обучающие виртуальные системы на основе онтологий и трехмерной компьютерной графики // Сборник научных трудов 13-й Международной конференции «Образование и виртуальность -2011». – Харьков-Ялта: УАДО, 2011. № 13. – С. 103-111.
44. Гулич С., Гундаварам Ш., Бирзнекс Г. CGI программирование на Perl – Пер. с англ. – СПб: Символ-Плюс, 2001. – 480 с.
45. Дзюбенко О.Л., Коженков А.О. Применение виртуальных симуляторов в образовательно-обучающей среде военного ВУЗа // Гуманитарные научные исследования. – № 1. – 2013 [Электронный ресурс]. URL: http://human.snauka.ru/2013/01/2181
46. Дзюбенко О.Л., Коженков А.О. Применение виртуальных симуляторов в обучении курсантов военного ВУЗа. // Психология, социология и педагогика. – 2012 [Электронный ресурс]. URL: http://psychology.snauka.ru/2012/07/942
47. Дозорцев В. М. Компьютерные тренажеры для обучения операторов технологических процессов - теория, методология построения и использования : автореферат диссертации на соискание ученой степени д-ра техн. наук:05.13.01, 05.13.06 / В. М. Дозорцев. - М., 1999. - 43 с.
48. Зубов М. Е. Математическое и программное обеспечение новых технологий проектирования виртуальных тренажеров: автореферат

диссертации на соискание ученой степени канд. техн. наук :05.13.11 / М. Е. Зубов. - М., 2003. - 17 с.

49. Ильин, А. М. Технология построения математического и программного обеспечения генерации окружающей обстановки для тренажерных комплексов: автореферат диссертации на соискание ученой степени канд. техн. наук :05.13.11 / А. М. Ильин. - Тула, 2008. - 20 с.
50. Катыс Г.П., Катыс П.Г. Виртуальная реальность в компьютерном обучении (часть1) // Дистанционное образование. - 1999. - N2.-С.17-22.
51. Катыс Г.П., Катыс П.Г Виртуальная реальность в компьютерном обучении (часть2) //Дистанционное образование. - 1999. - N3.-С.19-22. - С. 1999
52. Клещев А.С. Роль онтологии в программировании. Ч. 1. Аналитика // Информационные технологии. – 2008. – №10. – С. 42 – 46
53. Клещев А.С. Роль онтологии в программировании. Ч. 2. Интерактивное проектирование информационных объектов // Информационные технологии. – 2008. – №11. – С. 28 – 33
54. Князева, М. Д. Методы проектирования математических моделей и математического обеспечения для компьютерных систем и тренажеров: автореферат диссертации на соискание ученой степени канд. техн. наук: 05.13.01, 05.13.11 / М. Д. Князева. – М. – 1999. – 20 с.
55. Конверс Т., Парк Д., Морган К., PHP5 и MySQL. Библия пользователя. : Пер. с англ. – М. : Издательский дом «Вильямс». – 2006. – 1216 с.
56. Концер Т. «Облачные» вычисления: всё как сервис // PC Week/RE. №32 (638). [Элекстронный ресурс]. URL: http://www.pcweek.ru/themes/detail.php?ID=112879 (дата обращения: 27.09.2010)
57. Кузнецов М.В., Симдянов И.В., Голышев С.В. PHP5 на примерах. – СПб.: БХВ-Петербург, 2005. – 576 с.: ил.

58. Купцевич Ю.Е. Альманах программиста. ASP.Net, Web-сервисы, Web-приложения. – Москва: Русская редакция, 2003. – 401 с.
59. Левшин, С. А. Математическое и программное обеспечение для разработки специализированных вычислительных систем мобильных тренажеров: автореф. дис. на соиск. ученой степ. канд. техн. наук :05.13.18 / С. А. Левшин. – Новочеркасск. – 2009. –18 с.
60. Лотт Д., Шалл Д., Питерс К. ActionScript 3.0 Сборник рецептов. – Пер. с англ. – СПб: Символ-Плюс, 2007. – 608 с., ил.
61. Мещерякова М.А.. Учебный процесс вуза в системе управления качеством профессиональной подготовки врачей. Монография. М.: КДУ, 2006. - 140 с.
62. Миланина М. VRML в примерах. [Электронный ресурс]. URL: http://citforum.ru/internet/vrml/ (дата обращения: 10.06.2012)
63. Мук К. ActionScript 3.0 для Flash. Подробное руководство. – СПб.: Питер, 2009. – 992 с.: ил.
64. Новинки симуляционных технологий // Виртуальные технологии в медицине. – N1 (9). – 2013 [Электронный ресурс] –– URL: http://www.medsim.ru/file/2013-1/news-simulation-technologies.pdf
65. Петряева М.В., Федорищев Л.А. Формализация методов исследования в неврологии для медицинских интеллектуальных систем // Материалы Всероссийской научной Интернет-конференцит с международным участием «Современные системы искусственного интеллекта и их приложения в науке». – Казань: ИП Синяев Д.Н., 2013. – С.62-65.
66. Поленц Ф. и др. Виртуальная хирургия как новый метод обучения в стоматологической практике // Виртуальные технологии в медицине, №2 (8), 2012 [Электронный ресурс] –– URL: http://www.medsim.ru/file/2012-2/virtualniy_training_stomatologii.pdf
67. Проект «Авиация» [Электронный ресурс] –– URL: http://www.eligovision.ru/casestudy/9/

68. Проект «Салют» [Электронный ресурс] –– URL: http://www.eligovision.ru/casestudy/7/
69. Синецкий Р. М. Структурно-аппроксимационные методы распознавания речевых образов и их применение в тренажерно-моделирующих системах : автореф. дис. на соиск. ученой степ. канд. техн. наук :05.13.01 / Р. М. Синецкий. - Новочеркасск, 2008. - 30с.
70. Соснин П.И. Создание и использование онтологий проектов в разработке автоматизированных систем // КИИ 2010, Тверь. – 2010. – том 2. – С. 187-195
71. Стиренко А.С. 3ds Max 2009/3ds Max Design 2009. Самоучитель. – М.: ДМК Пресс. – 2008. – 544 с., ил.
72. Страуструп Б. Язык программирования С++. – М.: Бином. – 2008. – 1104 с.
73. Сук А.Ф. Виртуальная реальность и экспертные системы в дистанционном обучении // Інтелектуальні системи в промисловості і освіті - 2007 : тези доповідей Першої міжнародної науково-технічної конференції, 7-9 листопада 2007 / Відпов. за вип. А.С. Довбиш. - Суми : СумДУ, 2007. - С.62-63
74. Тимофеева А. Виртуальная реальность – через призму кинематографа. [Электронный ресурс]. URL: http://science.ua/publications/virtual-reality-in-cinema/ (дата обращения: 15.02.2011)
75. Томан М. Преимущества использования инструмента виртуальной реальности в судостроении // Материалы Международной конференции Моринтех-практик «Информационные технологии в судостроении-2012» [Электронный ресурс] –URL: http://www.remmag.ru/admin/upload_data/remmag/12-3/SENER.pdf
76. Торстейнсон П, Оберг Р. Архитектура .Net и программирование на Visual C++. : Пер. с англ. – М.: Издательский дом «Вильямс». – 2002. – 656 с.: ил.

77. Трухин А.В. Анализ существующих в РФ тренажерно-обучающих систем // Открытое и дистанционное образование. - 2008. - №1. - С. 32-39
78. Федорищев Л.А. Мультитекстурирование с помощью шейдеров // Программные продукты и системы. – 2013. – № 1. – С.58-61.
79. Федорищев Л.А. Применение шейдеров AGAL для задач текстурирования в 3D Интернет-приложениях // материалы XI Всероссийской научно-технической конференции «Теоретические и прикладные вопросы современных информационных технологий». – Улан-Удэ: Изд-во ВСГУТУ, 2012. – С. 96-100.
80. Филатова Н.Н., Ахремчик О.Л. Центр "Компьютерные технологии образования": его место в учебном процессе техническогоуниверситета.// Educational technology & Society. – 2000 – N1. – С.155-164. – [Электронный ресурс] – URL: http://ifets.ieee.org/russian/depository/v3_i2/html/5.html
81. Филатова Н.Н., Вавилова Н.И., Ахремчик О.JI. Виртуальные тренажеры как основа интенсификации когнитивных процессов // Материалы VII международной конференции "Современные технологии обучения". – Санкт-Петербург. – 2001. – Часть 1. –С. 200-202.
82. Филатова Н.Н., Вавилова Н.И. Представление знаний в мультимедиа тренажерах // Сборник научных трудов V международной научно-методической конференции "Новые информационные технологии в электротехническом образовании". – Астрахань, 2000. – С. 258-263. [Электронный ресурс] – URL: http://ckto.narod.ru/stastr.htm
83. Филатова Н.Н., Вавилова Н.И. Проектирование мультимедиа тренажеров на основе сценарных моделей представления знаний //Educational Technology S Society 3(4), 2000. – P.193-202.

84. Хиксон Я., HTML Is the New HTML5. [Электронный ресурс] – URL: http://blog.whatwg.org/html-is-the-new-html5/ (дата обращения: 25.01.2013)
85. Хилл Ф. OpenGL. Программирование компьютерной графики. Для профессионалов. – СПб.: Питер, 2002. – 1088 с.: ил.
86. Холодкова В. Виртуальная реальность: общие понятия, системы трекинга // Мир ПК, № 04, 2008 [Электронный ресурс] – URL: http://www.osp.ru/pcworld/2008/04/5175003/
87. Хураськин,И. А. , Методы и алгоритмы обработки визуальной информации для создания виртуального окружения тренажерных комплексов: автореферат диссертации на соискание ученой степени канд. техн. наук :05.13.01 / И. А. Хураськин. – М., 2008. – 20 с
88. Цыпцын С. и др. Maya. Сборник мастерклассов по продукту Autodesk|Maya от ведущих специалистов и дизайнеров России. – KostinPublishing, 2006. – 167 с.
89. Чижов А. Internet завтра в России и в мире: multimedia и виртуальная реальность. [Электронный ресурс]. URL: http://citforum.ru/internet/iinet96/21.shtml (дата обращения: 05.03.2013)
90. Шалфеева Е.А. Использование онтологий при проектировании решателей задач управляемых интеллектуальных систем // Материалы 4-й Всероссийской мультиконференции по проблемам управления (МКПУ-2011). – Таганрог: Изд-во ТТИ ЮФУ, – 2011. – Т. 1. – С. 167-170
91. Шаров В.В. Формирование профессиональных навыков эксплуатации САТОП с использованием компьютерной тренажерной системы АПА – 5Д: Дис. … канд. пед. наук. – Воронеж, 2000. – 184 с.
92. Шубина Л.Б. Мещерякова М.А., Камынина Н.Н., Уткина Г.Ю. Развитие медицинского образования в условиях инновационной экономики. [Электронный ресурс] – URL:

http://vestnik.mednet.ru/content/view/182/30/ (дата обращения: 04.04.2010)

93. Яговкин В. И. Разработка интерактивных тренажерных комплексов для освоения компетенций на основе предметных онтологий : автореф. дис. на соиск. ученой степ. канд. техн. наук :05.13.06 / В. И. Яговкин. - СПб., 2012. - 19 с.
94. Яркова С. А. Формирование информационного базиса автоматизированных систем обучения : автореферат диссертации на соискание ученой степени канд. техн. наук :05.13.01 / С. А. Яркова. - Красноярск, 2007. - 20 с.
95. Assenmacher I. et al. DataLaVista: A Packet-based Pipes and Filters Architecture for Data Handling in Virtual Environments, 3. Workshop der GI-Fachgruppe VR/AR, Koblenz [Electronic resource]. URL: http://www.rz.rwth-aachen.de/aw/cms/rz/Themen/Virtuelle_Realitaet/research/~piz/vr_software_vista/?lang=de (дата обращения: 16.01.2010)
96. Assenmacher I., Kuhlen T. The ViSTA Virtual Reality Toolkit, SEARIS Workhop on IEEE VR, Reno [Electronic resource]. URL: http://www.rz.rwth-aachen.de/aw/cms/rz/Themen/Virtuelle_Realitaet/research/~piz/vr_software_vista/?lang=de (дата обращения: 15.01.2010)
97. Balet O. et al. The CRIMSON Project: Simulating Populations in Massive Urban Environments // 8thWorld Congress on Computational Mechanles (WCCM8), 2008 [Electronic resource] – URL: http://www.crs4.it/vic/data/papers/wccm2008-crimson.pdf
98. Burdea G., Coiffet P., Virtual Reality Technology. – New York :Wiley & Sons, 2003. – 464 pp.
99. Craig A., Sherman W, Will J., Developing Virtual Reality Applications: Foundations of Effective Designpic .– New York: Morgan Kaufmann, 2009. – 448 pp.

100. Cruz-Neira C. et al. An Open Source Platform for Virtual Reality Applications Virtual Reality Applications Center Iowa State University [Electronic resource] — URL: http://oldsite.vrjuggler.org/pub/vrjuggler-aiaa2002.pdf

101. Evans C. phpIarchitect's Guide to Programming with Zend Framework. –Toronto: Marco Tabini&Associates Inc., 2008. – 222 pp.

102. Filatova N.N., Vavilova N.I. The trainer-simulator's models of the world on the basis of the plurality of the figurative representations // International Journal "Information Theories and Applications". – 2001. – volume 8, number 4. – pp. 176-184.

103. Filatova N., Vavilova N. The world's models of a trainer-simulator on the basis of figurative representations plurality // Information Theories and Applications. Varna. – 2000.

104. Fokkema J.T., Shuemie M.J., Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy [Electronic resource] — URL: http://graphics.tudelft.nl/~vrphobia/dissertation.pdf

105. Gray K. DirectX 9 Programmable Graphics Pipeline. – Washington: Microsoft Press, 2003. – 434 pp.

106. Gribova V. The concept of an intelligent tool for development of diagnostic computer simulators // Proceedings of First Russia and Pacific Conference on Computer Technology and Applications (RPC 2010). [Electronic res.]. –Vladivostok: IACP FEB RAS, 2010. – Pp.63-65

107. Gribova V.V., Fedorischev L.A. Internet Software Environment for Creating Teachware with Virtual Reality // Communications in Information Science and Management Engineering, World Academic Publishing Company, Hong Kong. 2012. – Vol. 2, – № 8. – Pp. 25-29 [Electronic resource] – URL: http://www.jcisme.org/Issue.aspx?Abstr=false

108. Gribova V.V., Fedorischev L.A. The architecture of Internet software environment for creating teachware with virtual reality // Emerging Intelligent Computing Technology and Applications 8th International

Conference (ICIC 2012). – Springer Berlin Heidelberg. 2012. – Vol. 304. – Part 11. – Pp. 394-399. – http://www.springerlink.com/content/g7k7p6542350u57w/

109. GoldStone W. Unity Game Development Essentials. – Birningham: Packt Publishing, 2009. – 316 pp.

110. Hansen K.. The Design of Public Space in 3D Virtual Worlds on the Internet. Virtual Space: Spatiality in Virtual Inhabited 3d Worlds. Lars Qvortrup, ed. London: Springer-Verlag. – 2002. – pp. 171 - 189

111. Herring C. Viable Software: the Intelligent Control Paradigm for Adaptable and Adaptive Architecture PhD Thesis, School of Information Technology and Electrical Engineering, The University of Queensland. – 2002 http://espace.library.uq.edu.au/view/UQ:157886.

112. Hirematada P., Flash 10 Multiplayer Game Essentials. – Birningham: Packt Publishing, – 2010. – 336 pp.

113. Jacko J., Sears A., Erlbaum L. The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications.– Mahwah Nj: Lawrence Erlbaum Associates. – 2003. – pp. 504-522

114. Johnson S. Adobe Flash Professional CS5 on demand. – Perspection Inc., 2010, – 577 p.

115. Kreylos O. This is a post about Vrui [Electronic resource] –– URL: http://doc-ok.org/?p=631

116. Kuhlen T., Beer T., Gerndt A. The ViSTA Virtual Reality Toolkit. 5th High-End Visualization Workshop, Baton Rouge, Lousianna [Electronic resource] –– URL: http://www.rz.rwthaachen.de/aw/cms/rz/Themen/Virtuelle_Realitaet/research/~piz/vr_software_vista/?lang=de

117. Lawson L., Brabant J., VRWorx2 Instructor Manual [Electronic resource] –– URL: http://uits.arizona.edu/sites/default/files/VRWorx_Instructor.pdf

118. Lecrenski N., Silverlight 4 Problem-Design-Solution. – Indianapolis: Wiley Publishing Inc., 2010. – 532 p.

119. Lukka T. An Introduction to VRML [Electronic resource] –– URL: (http://www.linuxjournal.com/article/3085)

120. Moline J., Virtual Reality in Health Care: a survey [Electronic resource] –– URL: http://www.cybertherapy.info/pages/survey.htm

121. Morganti F. (2004) Virtual interaction in cognitive neuropsychology. Studies in Health Technology and Informatics. – vol. 99. – p. 55-70 [Electronic resource] –– URL: http://www.morganti_studies_in_health_2004.pdf

122. Pecheny A., New Features of Alternativa3D 8.27 // Game development gems – software developer's journal 4/2012 [Electronic resource] –– URL: http://sdjournal.org/game-development-gems-software-developers-journal-42012-4/

123. Reyes R. Is HTML5 Replacing Flash . [Electronic resource] –– URL: http://www.lyquix.com/blog-and-news/is-html5-replacing-flash (дата обращения: 20.10.2012)

124. Riva G, Botella C, Légeron P, Optale (Eds.) Cybertherapy: Internet and Virtual Reality As Assessment and Rehabilitation Tools for Clinical Psychology and Neuroscience. – Amsterdam: Ios Press, – 2004.

125. Riva G., Design of clinical-oriented virtual environments: a communicational approach // CyberPsychology and Behavior, 2000. – pp. 351-357 [Electronic resource] –– URL: http://www.cybertherapy.info/pages/design.pdf

126. Riva G., Galimberti C., The Psychology of Cyberspace: a socio-cognitive framework to computer-mediated communication // New Ideas in Psychology, 15 (2), pp. 141-158, 1997 [Electronic resource] –– URL: http://www.cybertherapy.info/pages/cyber.htm

127. Riva G., Virtual Reality as communication tool: a socio-cognitive analysis // Journal Presence: Teleoperators and Virtual Environments. – V. 8, Issue 4. – 1999. – Pp. 462-468. [Electronic resource] — URL: http://www.neurovr.org/emerging/book1/1CHAPT_03.PDF

128. Riva G., Virtual Reality in psychotherapy: Review // CyberPsychology & Behavior. – V. 8, N. 3. – 2005. – Pp.220-240. [Electronic resource] — URL: http://www.cybertherapy.info/VR%20in%20psychotherapy.pdf

129. Rose F.D., et al. Virtual Reality in Brain Damage Rehabilitation: Review // CyberPsychology & Behavior. – V. 8, N. 3. – 2005. – Pp. 241-262. [Electronic resource] — URL: http://www.usc.edu/schools/medicine/departments/cell_neurobiology/research/isnsr/rizzo_docs/09_CyberPsychology_and_Behav_Rose_et_al.pdf

130. Satava R.M., Jones S.B., Medical application of Virtual Reality [Electronic resource] — URL: http://www.neurovr.org/pdf/papers/VR_Clinical/MedicalVR.pdf

131. Stanney K., Hale K. Handbook of Virtual Environments: Design, Implementation, and Applications. New Jersey: Lawrence Erlbaum. – 2002.

132. VE-Group Технологии виртуальной реальности [Electronic resource] — URL: http://ve-group.ru/

133. Venema S., System Design Issues for Virtual Reality simulations with haptic displays [Electronic resource] — URL: http://www.cybertherapy.info/pages/haptic.pdf (дата обращения: 07.03.2010)

134. Vincenti G., Multi-User Virtual Environments for the Classroom Practical Approaches to Teaching in Virtual Worlds . – IGI Global, 2011. [Electronic resource] — URL: http://eprints.adm.unipi.it/1219/1/LearningbyBuilding.pdf

135. Virtualization and Forensics: A Digital Forensic Investigator's Guide to Virtual Environments [Electronic resource] — URL: http://www.amazon.com/Virtualization-Forensics-Forensic-Investigators-Environments/dp/1597495573 (дата обращения: 17.03.2010)

136. Weiss P., Jessel A.S., Virtual Reality Applications to Work // Elsevier. – V. 11, N. 3. – 1998.– pp. 277-293

137. Yard T., Foundation ActionScript 3.0 Image Effects. – New York: Apress, 2009. – 684 p.

138. Zogrim, Popular Physics Engines comparison: PhysX, Havok and ODE [Electronic resource] — URL: http://physxinfo.com/articles/?page_id=154

ПРИЛОЖЕНИЕ 1

Классы и функции графического редактора

Группа классов виртуальной сцены

BaseModel. Базовый класс для объекта на виртуальной сцене. Содержит функции:

createMesh() – создание сетки трехмерного объекта, распарсивание бинарных данных из файла модели

initMaterials() - инициализация текстурных материалов на сетку объекта.

initOmniShadows() – инициализация теней от объекта от точечных источников света на сцене

initDirectShadows() – инициализация теней от объекта от прямых источников света

createAnimation() – инициализация скелетной анимации

createAnimationControllers() – создание контроллеров управления загруженной анимации объекта, позволяет вызывать нужные последовательности анимации в нужное время

addEvent() – функция добавления событий на объект

update() – функция обновления объекта. Если объект анимированный, выполняет переход на следующий кадр анимации

getState() – функция, возвращающая состояние изменяемого объекта

updateObjectState() – функция, обновляющая состояние изменяемого объекта на новое, заданное в качестве параметра функции.

stateTimer_timer() – функция, выполняющая анимацию объекта по таймеру (необходимо для анимационных переходов объектов)

setObjectStatePresent() – функция, устанавливающая презентационные атрибуты объекта.

setSprite() – функция инициализации спрайта

setAnimSprite() – функция инициализации анимированного спрайта

setAnimSpriteKeys() – функция для установки ключевых кадров анимированного спрайта

TextureManager. Класс управления текстурами.

find() – функция поиска текстуры. Если текстура имеется в КЭШе, то возвращается, иначе создается новая

recreate() – функция пересоздания текстуры в случае, если какие-то ее параметры изменились

createTex() – функция создания текстуры: создания для нее ресурса и загрузки в видеопамять компьютера

initTextures() – функция инициализации текстур указанной модели

AnimTextureManager. Класс управления анимированными текстурами.

find() – функция поиска текстуры. Если текстура имеется в КЭШе, то возвращается, иначе создается новая

createTex() – функция создания текстуры: создания для нее ресурса и загрузки в видеопамять компьютера

initClip() – функция создания раскадровки анимированной текстуры из заданного клипа анимации

createAnimTextures() – функция создания массива текстур для реализации анимации

ModelLoader. Класс для загрузки модели трехмерного объекта и ее ресурсов.

ModelLoader() – конструктор, устанавливающий параметры загрузки модели в соответствии с заданными по умолчанию и переданными в качестве входных данных.

load() – функция, выполняющая первичный запрос сервису о загрузке модели.

onLoadLibModel() – функция, выполняющаяся асинхронно после получения ответа от сервиса в результате выполнения функции load(). Получает и обрабатывает данные от сервиса о запрошенной модели: названия файлов, их адреса. Вызывает следующие функции загрузки отдельных компонентов модели: сетки, текстур, иконки

meshLoad() – функция загрузки мэша (сетки) модели.

onMeshLoad() – функция обратного вызова к функции meshLoad() после получения ответа от сервиса. После загрузки мэш представляет собой массив байт, который после обработки парсером 3d-моделей добавляется на виртуальную сцену методом scene.addChild().В случае если мэш и текстура не указаны для объекта, для него используются мэш и текстура по умолчанию (серый куб).

loadTexture() – функция загрузки текстуры модели

onTexLoad() – функция обратного вызова к функции loadTexture(). Мэш и текстура объекта кэшируются в клиенте, чтобы затем использовать их для эффективной инициализации других объектов, имеющих такую же структуру отображения.

initData() – функция, вызывающаяся после загрузки всех необходимых ресурсов модели для установки окончательных параметров. Устанавливаются презентационные атрибуты объекта, над объектом выводится информационный спрайт с названием этого объекта на сцене.

SceneLoader. Класс для загрузки сцены и ее элементов.

loadNewScene() – функция загрузки сцены по декларативной модели. Инициализирует рекурсивную функцию загрузки объектов сцены по дереву декларативной модели.

loadNode() – рекурсивная функция загрузки узла декларативной модели. Загружает

createOmniLight() – функция создания точечного источника света.

createCamera() – функция создания камеры

loadModel() – функция загрузки модели трехмерного объекта

onModelloaded() – функция обратного вызова к функции загрузки модели loadModel. Выполняет операции по вызову других инициализирующих функций и установке параметров на загруженную модель.

onEditorObjDown() – функция обратного вызова на событие нажатия мышкой по объекту сцены. Функция выполняет "выделение" объекта: перенос

координатных инструментов в центральную точку объекта, отображение параметров выделенного объекта в окне свойств.

SceneManager. Класс для управления сценой.

initData() - функция инициализации сцены по декларативной модели, загрузка ресурсов сцены в видеопамять компьютера, выделение текущего объекта и отображение сцены.

show() – функция отображения сцены.

loadObject() – функция загрузки нового объекта на сцену

updateObjectState() – функция обновления состояния указанного объекта сцены

doAction() – функция выполнения действия пользователя (команды или интерактивного взаимодействия с выбранным объектом). Делает необходимые проверки на выполнение, заданные в декларативной модели на данное действие, выполняет изменение состояний объектов, заданных в декларативной модели, отправляет запрос о выполнении действия на сервис.

onActionAnswer() – функция обратного вызова интерактивного взаимодействия пользователя с виртуальной сценой. Отображает сообщение о результате взаимодействия, изменяет указанные логические параметры объекта, вызывает следующие действия, если таковые имеются

findAttr() – функция поиска логического атрибута в заданного объекта

updateParams() – функция обновления логических параметров объекта.

delScene() – функция удаления сцены

changeModel() – функция изменения модели объекта. Новая модель применяется к выбранному объекту после выбора в библиотеке моделей.

getObjectByName() – функция получения созданного ранее объекта сцены по его имени

getActionByName() – функция получения действия по имени

loadActions() – функция инициализации действий из декларативной модели

update() – функция обновления всей сцены: обновление всех объектов (анимированных)

SpriteObject. Базовый класс для обьектов-спрайтов.

InfoSprite. Класс для информационных спрайтов – спрайтов, которые содержат тестовые сообщения на виртуальной сцене. Имеет только функцию – конструктор, создающий изображение из текста и на основе этого изображения – текстуру спрайта..

SpriteAnimObject. Класс для анимированных спрайтов.

SpriteAnimObject() – функция-конструктор, создающая из массива изображений анимированную текстуру спрайта.

update() – функция, вызываемая по таймеру для обновления кадров анимации текстуры спрайта.

EditManager. Класс управления редактором.

EditManager() – функция-конструктор, создающая все интерфейсные (включая 2d и 3d) элементы интерфейса редактора: модели стрелок для управления моделью, оконные формы библиотек моделей, меню редактора.

initObjectsMenu() – функция для создания списка объектов сцены в виде интерфейсного элемента для быстрого доступа к объектам.

onObjectChange() – функция обработки события выбора объекта из списка. Осуществляет выделение выбранного в списке объекта на сцене.

initMenu() – функция для инициализации меню редактора

onMenuChange() – функция обработки события выбора элемента меню. Вызов соответствующих меню функций.

loadLibsList() – функция для загрузки списка библиотек. Осуществляет запрос на сервис платформы.

onLoadLibsList() – функция для обработки получения списка библиотек от сервиса после вызова функции loadLibsList(). Вызывает инициализацию нового списка библиотек.

loadIco() – функция для загрузки иконки модели. Осуществляет запрос на иконку модели на сервис платформы.

onIcoLoad() - функция для обработки получения иконки модели от сервиса после вызова функции loadIco (). Вызывает отображение загруженной иконки.

open() – функция для открытия интерфейса редактора. Устанавливает на сцену визуальные инструменты редактирования (модели стрелок), вызывает функцию загрузки списка библиотек.

close() – функция для закрытия интерфейса редактора. Убирает со сцены визуальные инструменты редактирования.

toggle() – функция переключения открытия и закрытия интерфейса.

selectObject() – функция выделения объекта на виртуальной сцене. Осуществляет перенос виртуальных инструментов в центр выделенного объекта на сцене и отображает свойства выделенного объекта в окнах интерфейса (координаты, повороты и др.).

showObjectProperties() – функция отображения свойств выделенного объекта.

showObjectCoords() – функция отображения различных координат (в том числе, и поворотов, коэффициентов масштабирования) выделенного объекта.

setTool() – функция установки виртуальных инструментов управления в центр выделенного объекта.

saveScene() – функция для сохранения виртуальной сцены на сервисе платформы. Подготавливает в цикле все объекты сцены (собирает информацию о них, переводит в JSON-формат) и отправляет запрос на сервис платформы.

saveCurObject() – функция для сохранения текущего выделенного объекта. Собирает все данные о выделенном объекта в JSON-формате и отправляет соответствующий запрос о сохранении объекта на сервис платформы.

prepareObject() – функция подготовки объекта для сохранения на сервис. Функция подготовки данных объекта проходит по всем состояниям объекта, по всем логическим атрибутам в цикле и записывает их.

prepareStateObject() – функция подготовки состояния объекта для сохранения на сервис.

update() – функция обновления интерфейса. Выполняет обновление виртуальных инструментов и окна свойств объектов.

Группа классов инструментов

BaseToolModel. Базовый класс для модели инструмента.

initEvent() – инициализация событий на виртуальный инструмент

setToObject() – функция для размещения виртуального инструмента в центре выделенного объекта сцены. Устанавливает соответствующие координаты.

onDown() – функция обработки нажатия на виртуальный инструмент.move() – функция перемещения виртуального инструмента

update() – функция обновления виртуального инструмента. Актуальна, когда пользователь держит не отпуская клавиши управления данным инструментом, что позволяет обновлять его в режиме реального времени.

MoveToolModel. Класс инструмента перемещения.

MoveToolModel() – функция-конструктор, инициализирующая отображаемые виртуальные модели стрелок для инструмента перемещения объекта в пространстве.

move() – переопределенная функция перемещения, осуществляющая перемещение объекта и инструмента.

RotateToolModel. Класс инструмента вращения. *Аналогично*.

ScaleToolModel. Класс инструмента масштабирования. *Аналогично*.

Группа классов интерфейса

Interface. Класс-контейнер для контроля над всем интерфейсом редактора. Включает в свой состав экземпляры классов оконных форм.

EditorInterface.

SelectionFrame. Класс окна выбора библиотек, моделей.

ModelFrame. Класс, отображающий все модели текущей библиотеки.

BaseFrame. Базовый класс для оконных форм.

PropertiesFrame. Класс окна свойств объекта.

CoordsFrame. Класс окна координат объекта на сцене.

MessageFrame. Класс окна сообщений.

HistoryFrame. Класс окна истории.

Группы утилитных классов

FileLoader. Класс загрузки файлов с сервера платформы.

IcoLoader. Класс загрузки иконок.

SoundLoader. Класс загрузки звуков.

Global. Класс статических переменных.

Graphic. Класс функций с двумерной графикой.

Utils. Класс вспомогательных статических функций.

History. Класс управления историей действий пользователя.

ServerApi. Класс для взаимодействия с сервисом на платформе.

Классы и функции клиента интерпретатора

Группа классов виртуальной сцены

BaseModel. Базовый класс для объекта на виртуальной сцене.

TextureManager. Класс управления текстурами.

AnimTextureManager. Класс управления анимированными текстурами.

ModelLoader. Класс для загрузки модели трехмерного объекта и ее ресурсов.

SceneLoader. Класс для загрузки сцены и ее элементов.

SceneManager. Класс для управления сценой.

SpriteObject. Базовый класс для обьектов-спрайтов.

InfoSprite. Класс для информационных спрайтов – спрайтов, которые содержат тестовые сообщения на виртуальной сцене.

SpriteAnimObject. Класс для анимированных спрайтов.

Группа классов интерфейса

Interface. Класс-контейнер для контроля над всем интерфейсом редактора. Включает в свой состав экземпляры классов оконных форм.

ApplicationInterface. Класс интерфейса приложения интерпретатора.

BaseFrame. Базовый класс для оконных форм.

MessageFrame. Класс окна сообщений.

HistoryFrame. Класс окна истории.

NewHistoryFrame. Класс создания новой истории.

LogicDataFrame. Класс окна для отображения логических данных виртуальной среды.

Группы утилитных классов

FileLoader. Класс загрузки файлов с сервера платформы.

IcoLoader. Класс загрузки иконок.

SoundLoader. Класс загрузки звуков.

Global. Класс статических переменных.

Graphic. Класс функций с двумерной графикой.

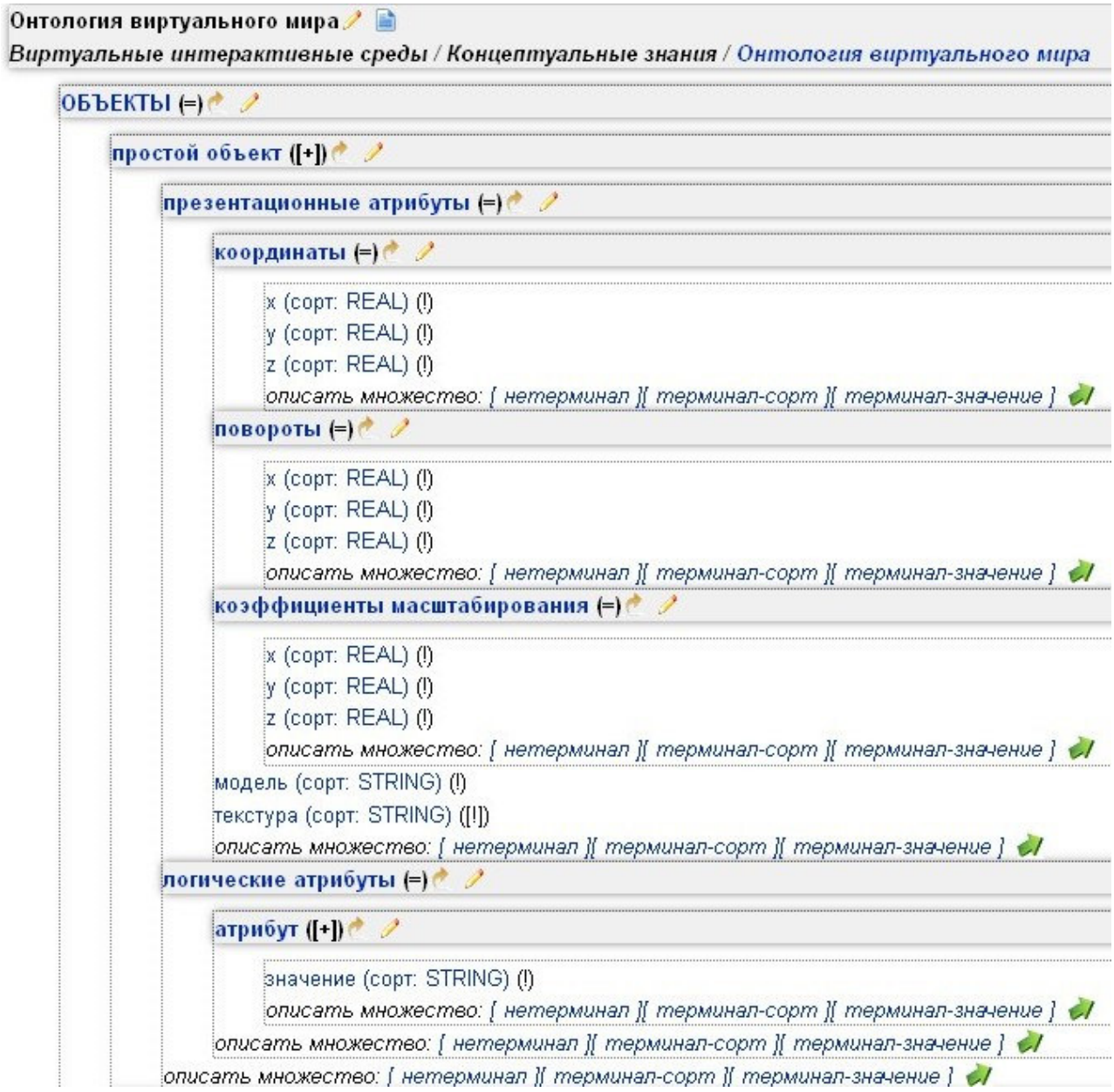
Utils. Класс вспомогательных статических функций.

History. Класс управления историей действий пользователя.

ServerApi. Класс для взаимодействия с сервисом на платформе.

ПРИЛОЖЕНИЕ 2

Реализация онтологии профессиональных виртуальных сред в структурном редакторе




изменяемый объект ([+])
-> логические атрибуты (=)
множество состояний (=)
состояние (+)
презентационные атрибуты (=)
-> координаты (=)
-> повороты (=)
-> коэффициенты масштабирования (=)
текстура (сорт: STRING) (!)
анимация (сорт: STRING) (!)
переход (=)
время (сорт: REAL) (!)
анимация (сорт: STRING) ([!])
интерполяция поворота (сорт: BOOLEAN) ([!])
описать множество: [нетерминал][терминал-сорт][терминал-значение]
модель (сорт: STRING) (!)
описать множество: [нетерминал][терминал-сорт][терминал-значение]
-> логические атрибуты (=)
описать множество: [нетерминал][терминал-сорт][терминал-значение]
описать множество: [нетерминал][терминал-сорт][терминал-значение]
-> презентационные атрибуты (=)
-> состояние (=)
описать множество: [нетерминал][терминал-сорт][терминал-значение]
таблица ([+])
-> презентационные атрибуты (=)
-> логические атрибуты (=)
элементы (=)
элемент (+)
презентационные атрибуты (=)
x1 (сорт: REAL) (!)
y1 (сорт: REAL) (!)
x2 (сорт: REAL) (!)
y2 (сорт: REAL) (!)
описать множество: [нетерминал][терминал-сорт][терминал-значение]
-> логические атрибуты (=)
описать множество: [нетерминал][терминал-сорт][терминал-значение]
описать множество: [нетерминал][терминал-сорт][терминал-значение]
описать множество: [нетерминал][терминал-сорт][терминал-значение]

источник света ([+])

- **презентационные атрибуты (=)**
  - -> координаты (=)
  - -> повороты (=)
  - -> коэффициенты масштабирования (=)
  - *описать множество: [нетерминал][терминал-сорт][терминал-значение]*
- камера (тип: STRING) (=)
- цвет (сорт: STRING) (!)
- **тип (=)**
  - точечный (тип: STRING) (=)
  - направленный (тип: STRING) (=)
  - *описать множество альтернатив: [нетерминал][терминал-сорт][терминал-значение]*
- множество состояний (=)
- активность (сорт: BOOLEAN) (!)
- **состояние (=)**
  - -> презентационные атрибуты (=)
  - активность (сорт: BOOLEAN) (!)
  - *описать множество: [нетерминал][терминал-сорт][терминал-значение]*
- интенсивность (сорт: REAL) (!)
- тень (сорт: BOOLEAN) (!)
- *описать множество: [нетерминал][терминал-сорт][терминал-значение]*

камера ([+])

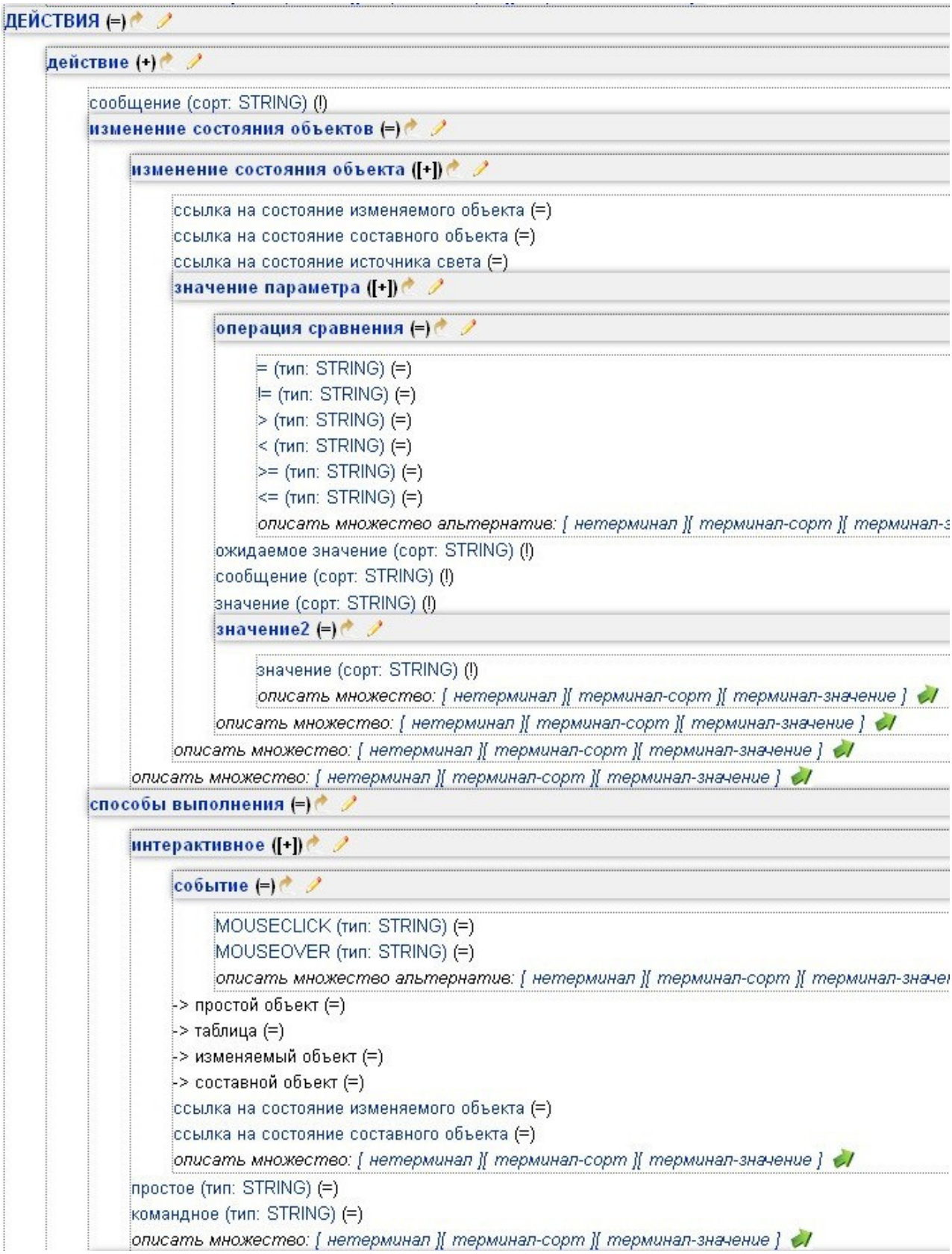
- -> презентационные атрибуты (=)
- *описать множество: [нетерминал][терминал-сорт][терминал-значение]*

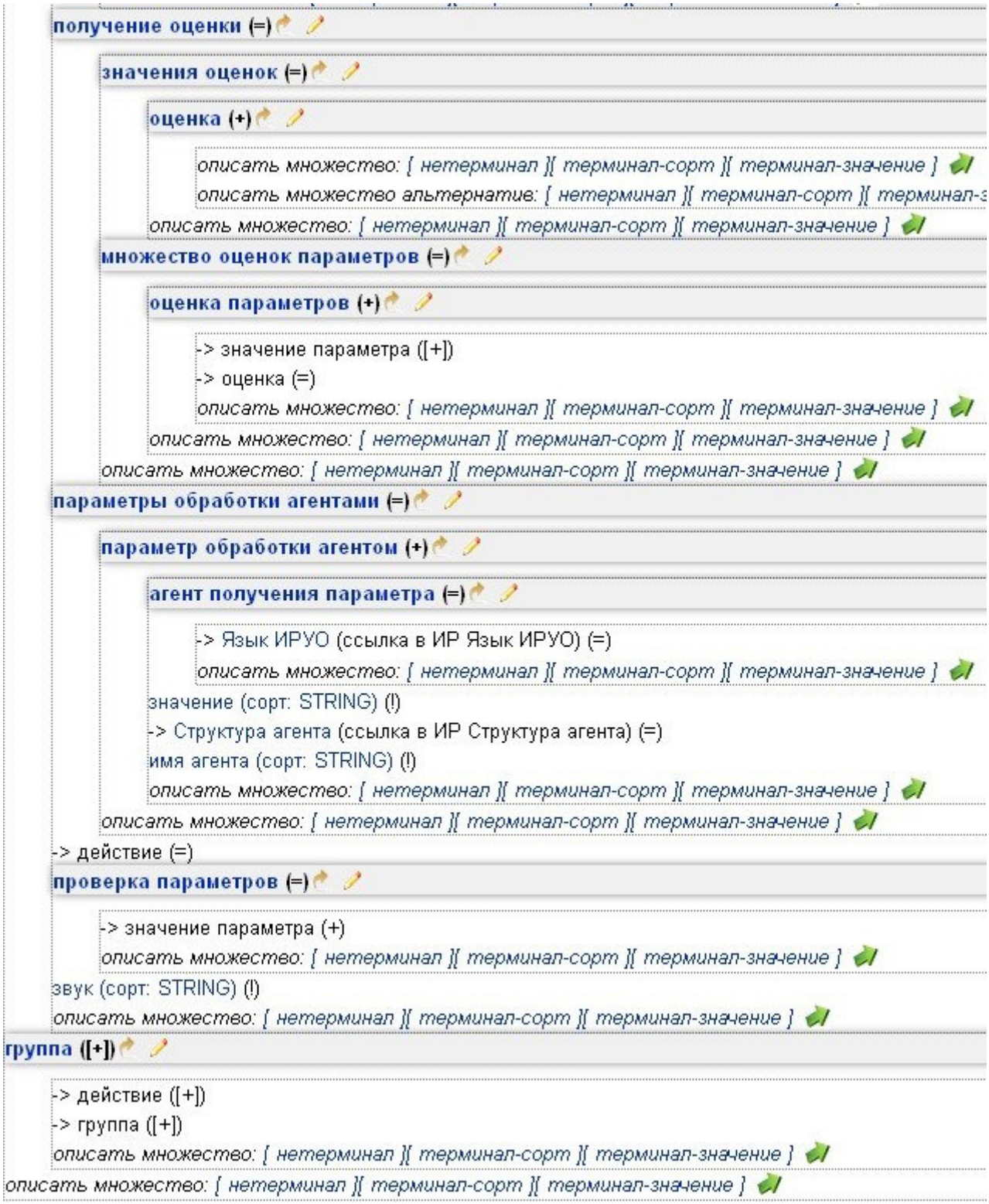
составной объект ([+])

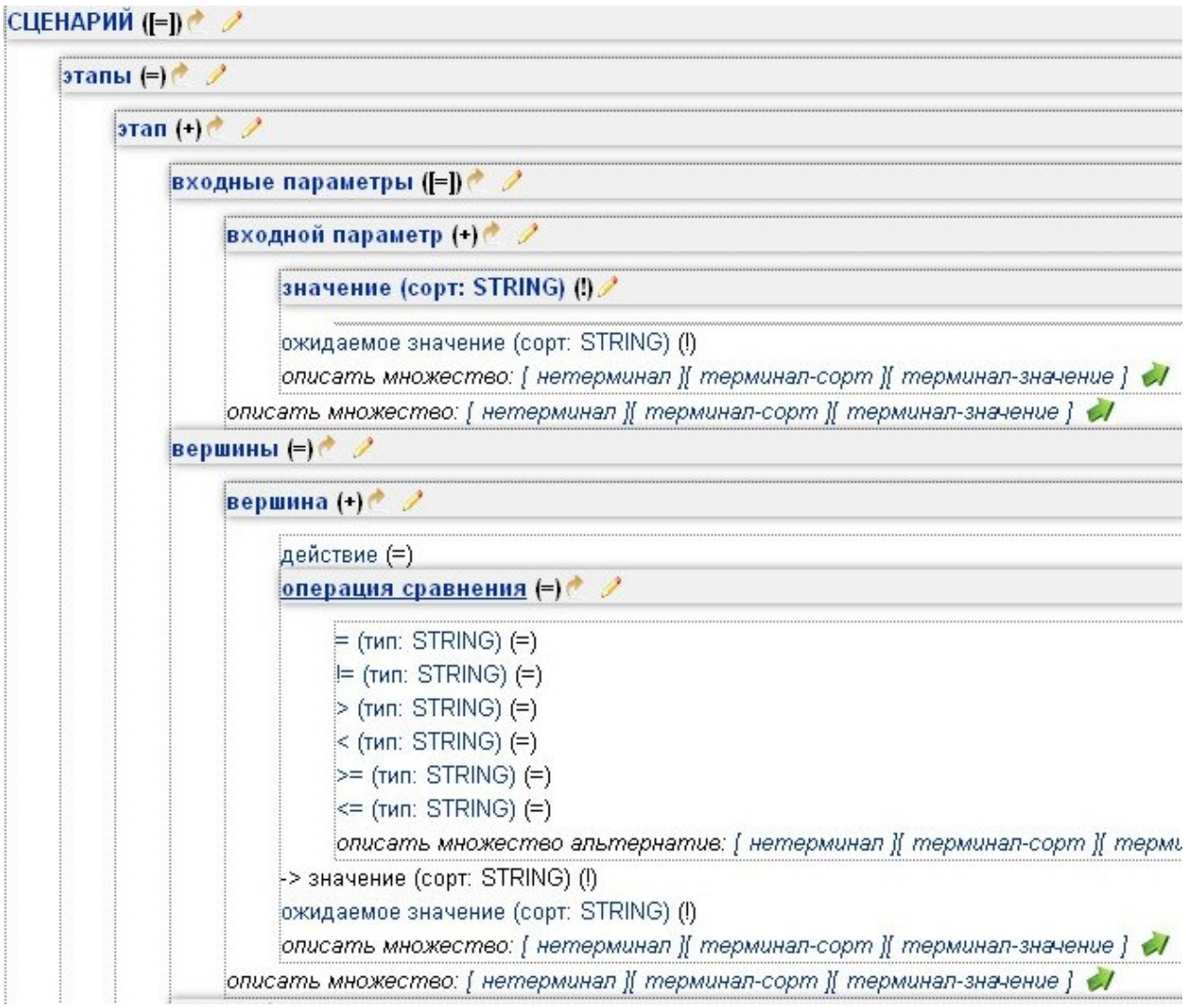
- -> логические атрибуты (=)
- множество состояний (=)
- -> презентационные атрибуты (=)
- дочерние объекты (=)
- состояние (=)
- *описать множество: [нетерминал][терминал-сорт][терминал-значение]*

группа ([+])

- -> простой объект ([+])
- -> изменяемый объект ([+])
- -> таблица ([+])
- -> составной объект ([+])
- -> источник света ([+])
- -> камера ([+])
- -> группа ([+])
- *описать множество: [нетерминал][терминал-сорт][терминал-значение]*

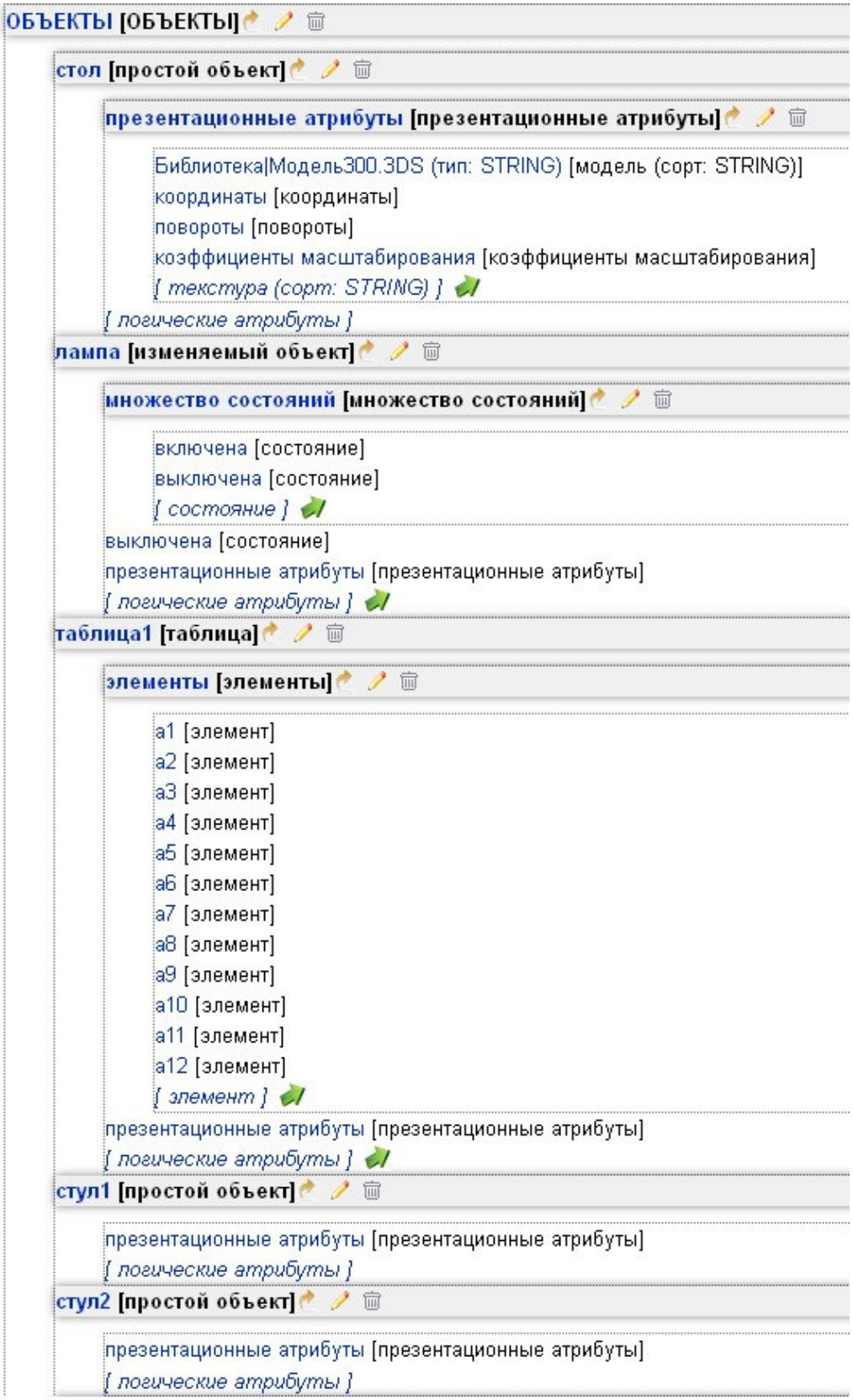

ДЕЙСТВИЯ (=)
действие (+)
сообщение (сорт: STRING) (!)
изменение состояния объектов (=)
изменение состояния объекта ([+])
ссылка на состояние изменяемого объекта (=)
ссылка на состояние составного объекта (=)
ссылка на состояние источника света (=)
значение параметра ([+])
операция сравнения (=)
= (тип: STRING) (=)
!= (тип: STRING) (=)
> (тип: STRING) (=)
< (тип: STRING) (=)
>= (тип: STRING) (=)
<= (тип: STRING) (=)
описать множество альтернатив: [нетерминал][терминал-сорт][терминал-
ожидаемое значение (сорт: STRING) (!)
сообщение (сорт: STRING) (!)
значение (сорт: STRING) (!)
значение2 (=)
значение (сорт: STRING) (!)
описать множество: [нетерминал][терминал-сорт][терминал-значение]
описать множество: [нетерминал][терминал-сорт][терминал-значение]
описать множество: [нетерминал][терминал-сорт][терминал-значение]
описать множество: [нетерминал][терминал-сорт][терминал-значение]
способы выполнения (=)
интерактивное ([+])
событие (=)
MOUSECLICK (тип: STRING) (=)
MOUSEOVER (тип: STRING) (=)
описать множество альтернатив: [нетерминал][терминал-сорт][терминал-значе
-> простой объект (=)
-> таблица (=)
-> изменяемый объект (=)
-> составной объект (=)
ссылка на состояние изменяемого объекта (=)
ссылка на состояние составного объекта (=)
описать множество: [нетерминал][терминал-сорт][терминал-значение]
простое (тип: STRING) (=)
командное (тип: STRING) (=)
описать множество: [нетерминал][терминал-сорт][терминал-значение]


получение оценки (=)
значения оценок (=)
оценка (+)
описать множество: [нетерминал][терминал-сорт][терминал-значение]
описать множество альтернатив: [нетерминал][терминал-сорт][терминал-
описать множество: [нетерминал][терминал-сорт][терминал-значение]
множество оценок параметров (=)
оценка параметров (+)
-> значение параметра ([+])
-> оценка (=)
описать множество: [нетерминал][терминал-сорт][терминал-значение]
описать множество: [нетерминал][терминал-сорт][терминал-значение]
описать множество: [нетерминал][терминал-сорт][терминал-значение]
параметры обработки агентами (=)
параметр обработки агентом (+)
агент получения параметра (=)
-> Язык ИРУО (ссылка в ИР Язык ИРУО) (=)
описать множество: [нетерминал][терминал-сорт][терминал-значение]
значение (сорт: STRING) (!)
-> Структура агента (ссылка в ИР Структура агента) (=)
имя агента (сорт: STRING) (!)
описать множество: [нетерминал][терминал-сорт][терминал-значение]
описать множество: [нетерминал][терминал-сорт][терминал-значение]
-> действие (=)
проверка параметров (=)
-> значение параметра (+)
описать множество: [нетерминал][терминал-сорт][терминал-значение]
звук (сорт: STRING) (!)
описать множество: [нетерминал][терминал-сорт][терминал-значение]
группа ([+])
-> действие ([+])
-> группа ([+])
описать множество: [нетерминал][терминал-сорт][терминал-значение]
описать множество: [нетерминал][терминал-сорт][терминал-значение]


СЦЕНАРИЙ ([=])
этапы (=)
этап (+)
входные параметры ([=])
входной параметр (+)
значение (сорт: STRING) (!)
ожидаемое значение (сорт: STRING) (!)
описать множество: [нетерминал][терминал-сорт][терминал-значение]
описать множество: [нетерминал][терминал-сорт][терминал-значение]
вершины (=)
вершина (+)
действие (=)
операция сравнения (=)
= (тип: STRING) (=)
!= (тип: STRING) (=)
> (тип: STRING) (=)
< (тип: STRING) (=)
>= (тип: STRING) (=)
<= (тип: STRING) (=)
описать множество альтернатив: [нетерминал][терминал-сорт][терми
-> значение (сорт: STRING) (!)
ожидаемое значение (сорт: STRING) (!)
описать множество: [нетерминал][терминал-сорт][терминал-значение]
описать множество: [нетерминал][терминал-сорт][терминал-значение]

ПРИЛОЖЕНИЕ 3

Реализация декларативной модели компьютерного тренажера для обучения проверке зрения по таблицам Сивцева в структурном редакторе



кровать [простой объект]

презентационные атрибуты [презентационные атрибуты]
[логические атрибуты]

мониторы [простой объект]

презентационные атрибуты [презентационные атрибуты]
[логические атрибуты]

прибор [простой объект]

презентационные атрибуты [презентационные атрибуты]
[логические атрибуты]

люстра [простой объект]

презентационные атрибуты [презентационные атрибуты]
[логические атрибуты]

комната [простой объект]

презентационные атрибуты [презентационные атрибуты]
[логические атрибуты]

аппарат Рота [изменяемый объект]

множество состояний [множество состояний]

включен [состояние]
выключен [состояние]
[состояние]

выключен [состояние]
презентационные атрибуты [презентационные атрибуты]
[логические атрибуты]

пациент [изменяемый объект]

множество состояний [множество состояний]

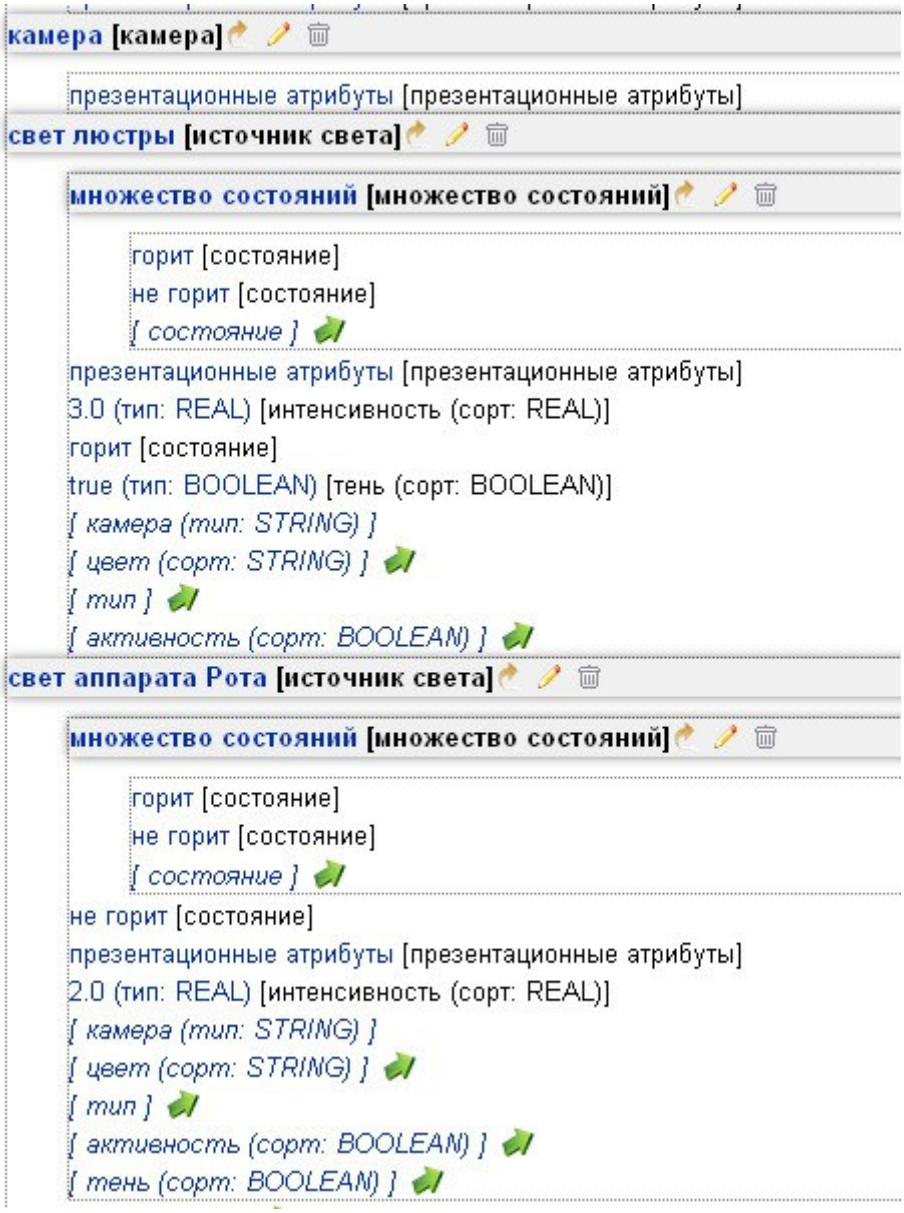
сидит1 [состояние]
закрывает правый глаз [состояние]
сидит2 [состояние]
закрывает левый глаз [состояние]
сидит3 [состояние]
[состояние]

сидит1 [состояние]
логические атрибуты [логические атрибуты]

острота [атрибут]

0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0 (тип: STRING) [значение (сорт: STRING)]
[атрибут]

презентационные атрибуты [презентационные атрибуты]


камера [камера]
презентационные атрибуты [презентационные атрибуты]
свет люстры [источник света]
множество состояний [множество состояний]
горит [состояние]
не горит [состояние]
[состояние]
презентационные атрибуты [презентационные атрибуты]
3.0 (тип: REAL) [интенсивность (сорт: REAL)]
горит [состояние]
true (тип: BOOLEAN) [тень (сорт: BOOLEAN)]
[камера (тип: STRING)]
[цвет (сорт: STRING)]
[тип]
[активность (сорт: BOOLEAN)]
свет аппарата Рота [источник света]
множество состояний [множество состояний]
горит [состояние]
не горит [состояние]
[состояние]
не горит [состояние]
презентационные атрибуты [презентационные атрибуты]
2.0 (тип: REAL) [интенсивность (сорт: REAL)]
[камера (тип: STRING)]
[цвет (сорт: STRING)]
[тип]
[активность (сорт: BOOLEAN)]
[тень (сорт: BOOLEAN)]

ДЕЙСТВИЯ [ДЕЙСТВИЯ]

назвать буквы [действие]

закрыть правый глаз [действие]

способы выполнения [способы выполнения]

командное (тип: STRING) [командное (тип: STRING)]

[интерактивное]

[простое (тип: STRING)]

изменение состояния объектов [изменение состояния объектов]

закрывание правого глаза [изменение состояния объекта]

ссылка на состояние изменяемого объекта [ссылка на состояние изменяемого

закрывает правый глаз [состояние]

[ссылка на состояние составного объекта]

[ссылка на состояние источника света]

[значение параметра]

[изменение состояния объекта]

Библиотека офт. табл.|закройте правый глаз (тип: STRING) [звук (сорт: STRING)]

[сообщение (сорт: STRING)]

[получение оценки]

[параметры обработки агентами]

[действие]

[проверка параметров]

включить аппарат Рота [действие]

способы выполнения [способы выполнения]

командное (тип: STRING) [командное (тип: STRING)]

[интерактивное]

[простое (тип: STRING)]

изменение состояния объектов [изменение состояния объектов]

включение аппарата Рота [изменение состояния объекта]

ссылка на состояние изменяемого объекта [ссылка на состояние изменяемого

включен [состояние]

[ссылка на состояние составного объекта]

[ссылка на состояние источника света]

[значение параметра]

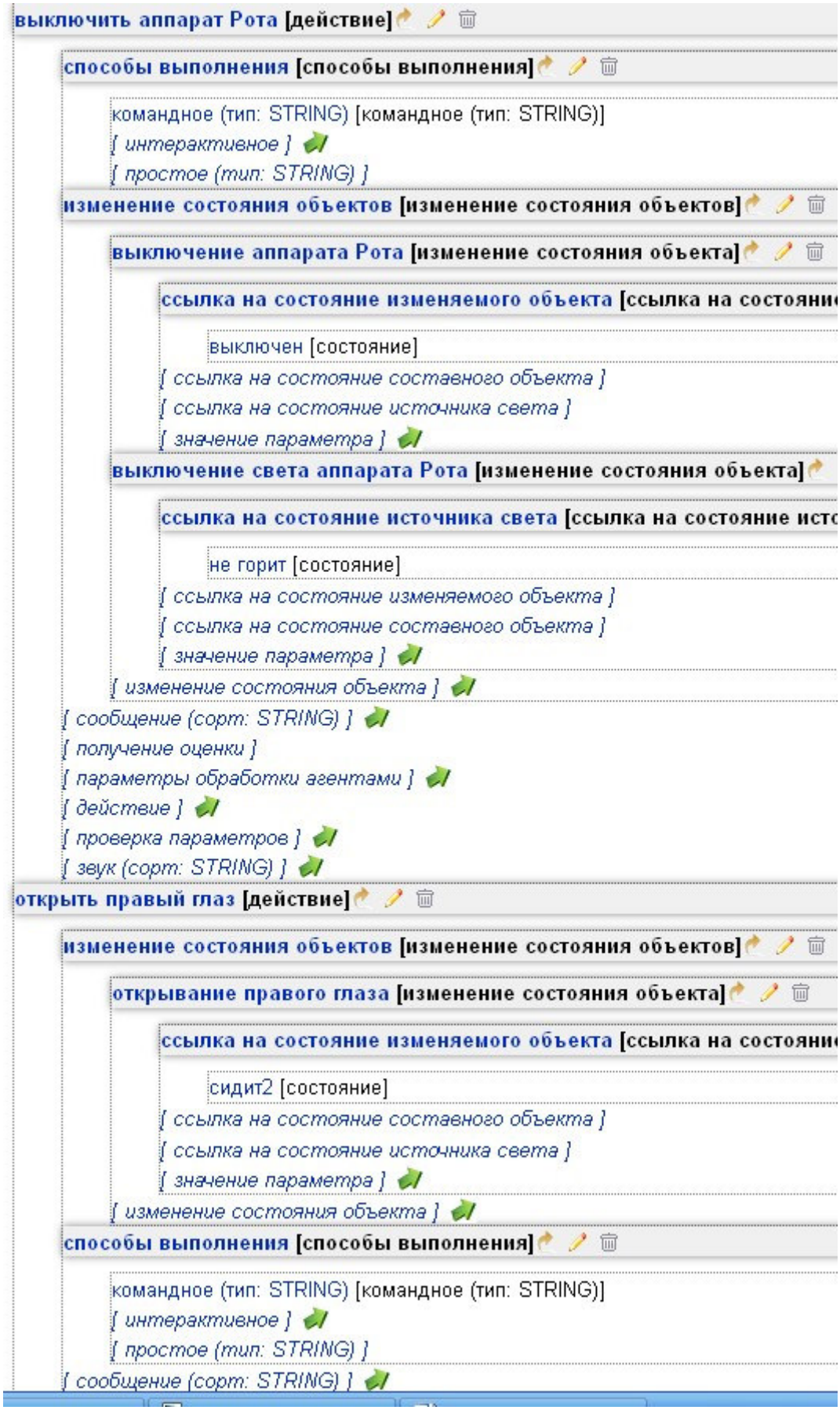
включение света аппарата Рота [изменение состояния объекта]

ссылка на состояние источника света [ссылка на состояние источника света]

горит [состояние]

[ссылка на состояние изменяемого объекта]

[ссылка на состояние составного объекта]


выключить аппарат Рота [действие]
способы выполнения [способы выполнения]
командное (тип: STRING) [командное (тип: STRING)]
[интерактивное]
[простое (тип: STRING)]
изменение состояния объектов [изменение состояния объектов]
выключение аппарата Рота [изменение состояния объекта]
ссылка на состояние изменяемого объекта [ссылка на состояни
выключен [состояние]
[ссылка на состояние составного объекта]
[ссылка на состояние источника света]
[значение параметра]
выключение света аппарата Рота [изменение состояния объекта]
ссылка на состояние источника света [ссылка на состояние исто
не горит [состояние]
[ссылка на состояние изменяемого объекта]
[ссылка на состояние составного объекта]
[значение параметра]
[изменение состояния объекта]
[сообщение (сорт: STRING)]
[получение оценки]
[параметры обработки агентами]
[действие]
[проверка параметров]
[звук (сорт: STRING)]
открыть правый глаз [действие]
изменение состояния объектов [изменение состояния объектов]
открывание правого глаза [изменение состояния объекта]
ссылка на состояние изменяемого объекта [ссылка на состояни
сидит2 [состояние]
[ссылка на состояние составного объекта]
[ссылка на состояние источника света]
[значение параметра]
[изменение состояния объекта]
способы выполнения [способы выполнения]
командное (тип: STRING) [командное (тип: STRING)]
[интерактивное]
[простое (тип: STRING)]
[сообщение (сорт: STRING)]


закрыть левый глаз [действие]
способы выполнения [способы выполнения]
командное (тип: STRING) [командное (тип: STRING)]
[интерактивное]
[простое (тип: STRING)]
изменение состояния объектов [изменение состояния объектов]
закрывание левого глаза [изменение состояния объекта]
ссылка на состояние изменяемого объекта [ссылка на состояние
закрывает левый глаз [состояние]
[ссылка на состояние составного объекта]
[ссылка на состояние источника света]
[значение параметра]
[изменение состояния объекта]
[сообщение (сорт: STRING)]
[получение оценки]
[параметры обработки агентами]
[действие]
[проверка параметров]
[звук (сорт: STRING)]
открыть левый глаз [действие]
способы выполнения [способы выполнения]
командное (тип: STRING) [командное (тип: STRING)]
[интерактивное]
[простое (тип: STRING)]
изменение состояния объектов [изменение состояния объектов]
открывание левого глаза [изменение состояния объекта]
ссылка на состояние изменяемого объекта [ссылка на состояние
сидит3 [состояние]
[ссылка на состояние составного объекта]
[ссылка на состояние источника света]
[значение параметра]
[изменение состояния объекта]
[сообщение (сорт: STRING)]
[получение оценки]
[параметры обработки агентами]
[действие]
[проверка параметров]
[звук (сорт: STRING)]


ДЕЙСТВИЯ [ДЕЙСТВИЯ]
назвать буквы [действие]
способы выполнения [способы выполнения]
мышкой [интерактивное]
событие [событие]
MOUSECLICK (тип: STRING) [MOUSECLICK (тип: STRING)]
таблица1 [таблица]
[простой объект]
[изменяемый объект]
[составной объект]
[ссылка на состояние изменяемого объекта]
[ссылка на состояние составного объекта]
[интерактивное]
[простое (тип: STRING)]
[командное (тип: STRING)]
параметры обработки агентами [параметры обработки агентами]
строка [параметр обработки агентом]
1 (тип: STRING) [значение (сорт: STRING)]
агент получения параметра [агент получения параметра]
Виртуальные интерактивные среды / Медицина / Агент обработки офта.
агента (сорт: STRING)]
[Структура агента]
[параметр обработки агентом]
получение оценки [получение оценки]
значения оценок [значения оценок]
0 ошибок [оценка]
1 ошибка [оценка]
2 ошибки [оценка]
>2 ошибок [оценка]
не видит [оценка]
[оценка]


множество оценок параметров [множество оценок параметров]
набор1 [оценка параметров]
строка [значение параметра]
1 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
острота [значение параметра]
0.1 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
-> 0 ошибок [оценка]
[значение параметра]
набор2 [оценка параметров]
строка [значение параметра]
1-2 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
-> острота [значение параметра]
-> 1 ошибка [оценка]
[значение параметра]
набор3 [оценка параметров]
строка [значение параметра]
2-3 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
-> острота [значение параметра]
-> 2 ошибки [оценка]
[значение параметра]

набор4 [оценка параметров]

строка [значение параметра]

2-3 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]

острота [значение параметра]

0.1 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]

-> >2 ошибок [оценка]
[значение параметра]

набор5 [оценка параметров]

строка [значение параметра]

3-12 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]

-> острота [значение параметра]
-> не видит [оценка]
[значение параметра]

набор6 [оценка параметров]

строка [значение параметра]

1-2 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]

острота [значение параметра]

0.2 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]

-> 0 ошибок [оценка]
[значение параметра]

набор7 **[оценка параметров]**

строка **[значение параметра]**

2-3 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]

острота **[значение параметра]**

0.2 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]

-> 1 ошибка [оценка]
[значение параметра]

набор8 **[оценка параметров]**

острота **[значение параметра]**

0.2 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]

-> 2 ошибки [оценка]

строка **[значение параметра]**

3-4 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]

[значение параметра]

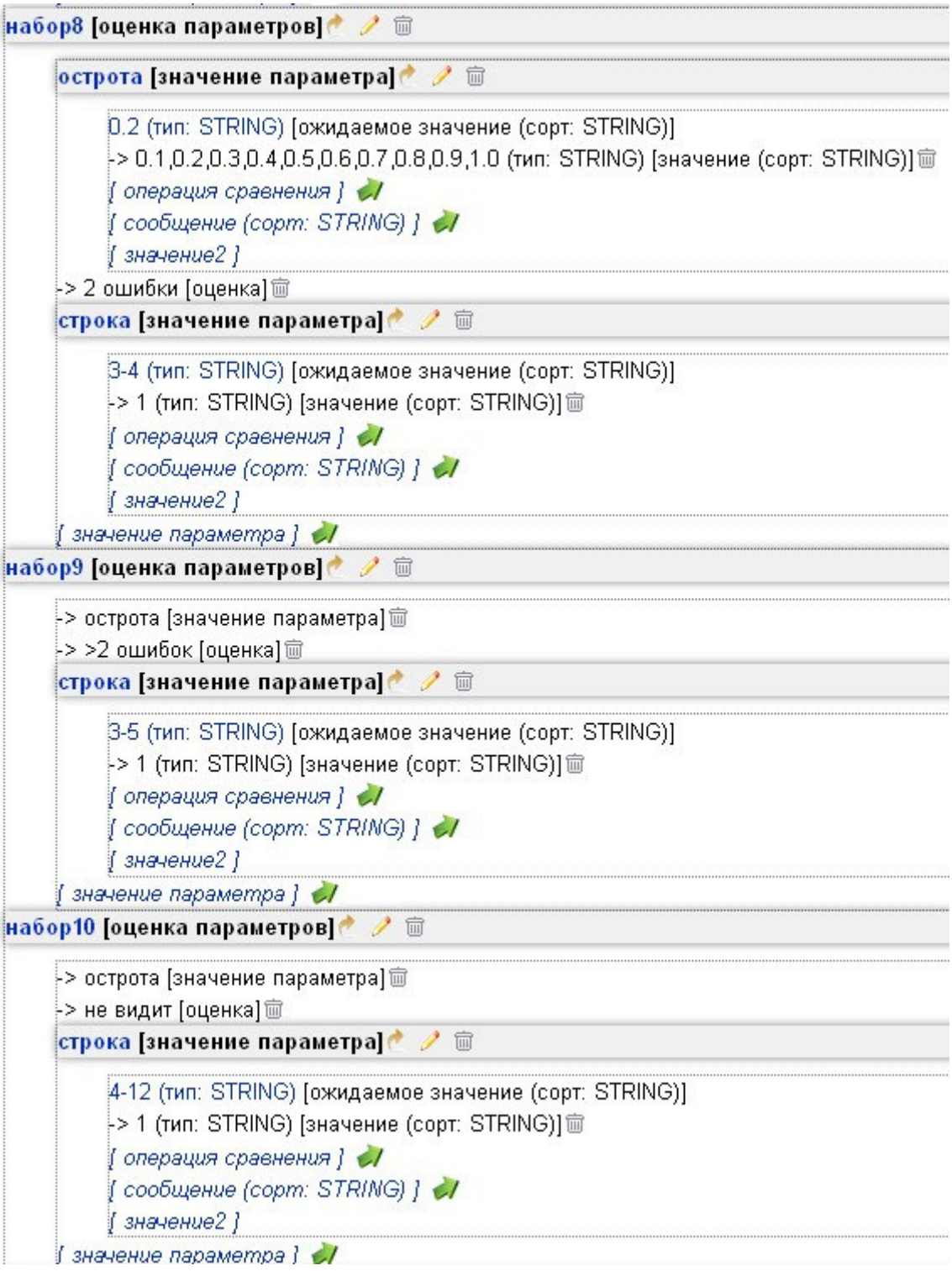
набор9 **[оценка параметров]**

-> острота [значение параметра]
-> >2 ошибок [оценка]

строка **[значение параметра]**

3-5 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]

[значение параметра]


набор8 [оценка параметров]
острота [значение параметра]
0.2 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
-> 2 ошибки [оценка]
строка [значение параметра]
3-4 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
[значение параметра]
набор9 [оценка параметров]
-> острота [значение параметра]
-> >2 ошибок [оценка]
строка [значение параметра]
3-5 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
[значение параметра]
набор10 [оценка параметров]
-> острота [значение параметра]
-> не видит [оценка]
строка [значение параметра]
4-12 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
[значение параметра]

набор11 **[оценка параметров]**

строка **[значение параметра]**

1-3 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]

острота **[значение параметра]**

0.3 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]

-> 0 ошибок [оценка]
[значение параметра]

набор12 **[оценка параметров]**

-> острота [значение параметра]
-> 1 ошибка [оценка]

строка **[значение параметра]**

3-4 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]

[значение параметра]

набор13 **[оценка параметров]**

строка **[значение параметра]**

4-5 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]

-> острота [значение параметра]
-> 2 ошибки [оценка]
[значение параметра]

набор14 [оценка параметров]

острота [значение параметра]

0.3 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]

-> >2 ошибок [оценка]

строка [значение параметра]

4-5 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]

[значение параметра]

набор15 [оценка параметров]

-> острота [значение параметра]
-> не видит [оценка]

строка [значение параметра]

5-12 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]

[значение параметра]

набор16 [оценка параметров]

-> 0 ошибок [оценка]

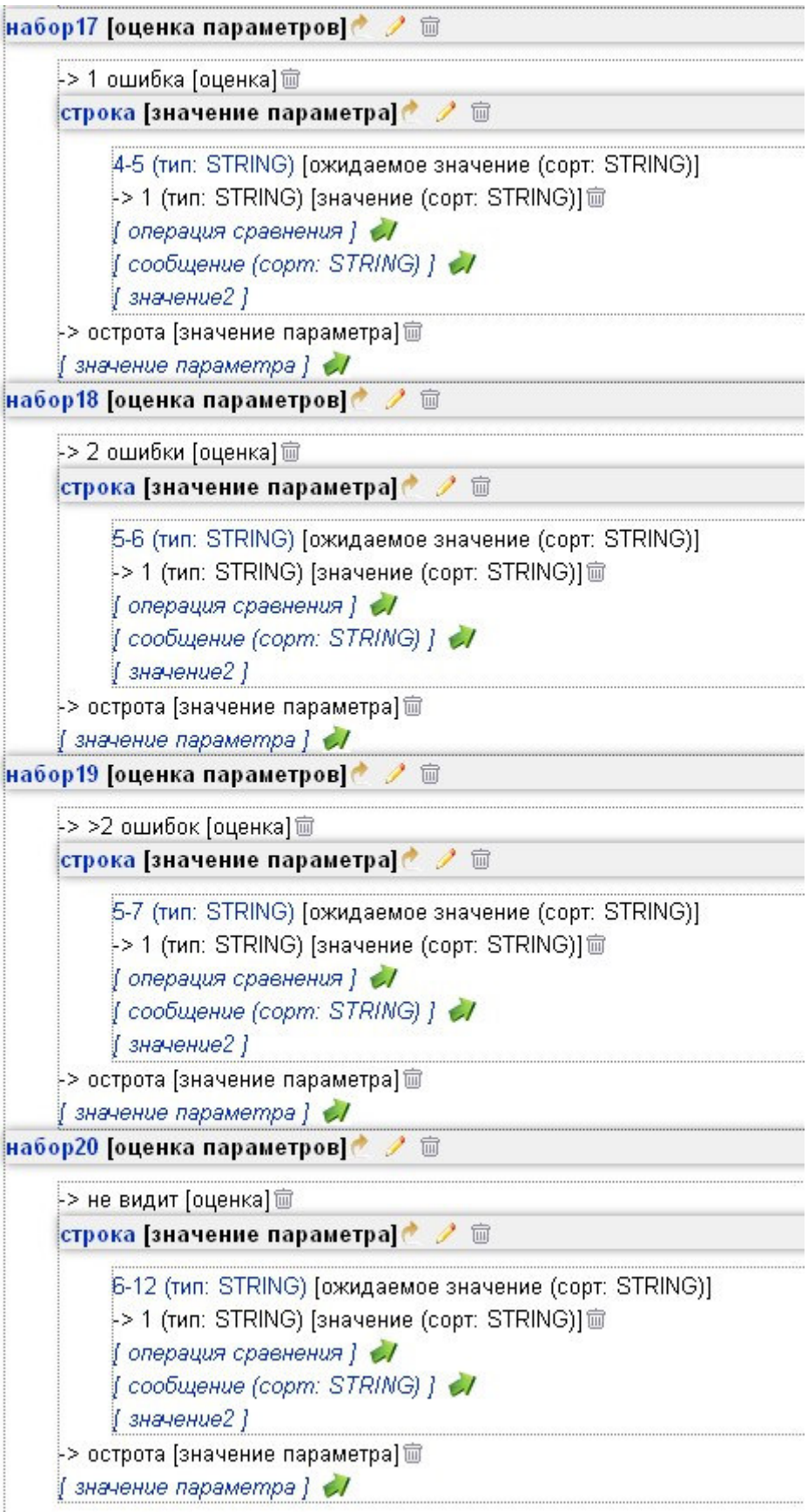
строка [значение параметра]

1-4 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]

острота [значение параметра]

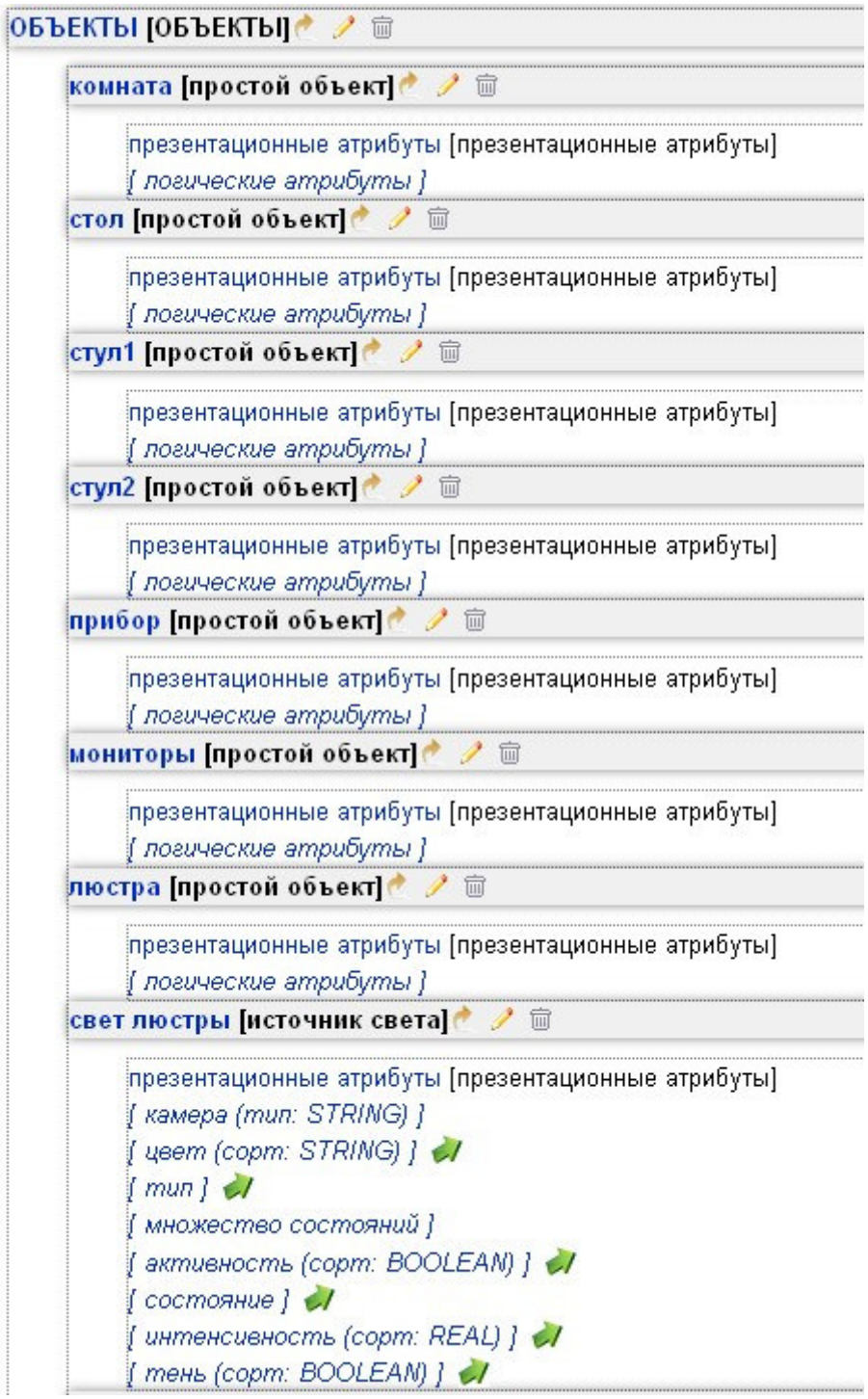
0.4 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]

[значение параметра]


набор17 [оценка параметров]
-> 1 ошибка [оценка]
строка [значение параметра]
4-5 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
-> острота [значение параметра]
[значение параметра]
набор18 [оценка параметров]
-> 2 ошибки [оценка]
строка [значение параметра]
5-6 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
-> острота [значение параметра]
[значение параметра]
набор19 [оценка параметров]
-> >2 ошибок [оценка]
строка [значение параметра]
5-7 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
-> острота [значение параметра]
[значение параметра]
набор20 [оценка параметров]
-> не видит [оценка]
строка [значение параметра]
6-12 (тип: STRING) [ожидаемое значение (сорт: STRING)]
-> 1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
-> острота [значение параметра]
[значение параметра]


СЦЕНАРИЙ [СЦЕНАРИЙ]
этапы [этапы]
этап1 [этап]
вершины [вершины]
включение аппарата Рота [вершина]
включить аппарат Рота [действие]
[операция сравнения]
[значение (сорт: STRING)]
[ожидаемое значение (сорт: STRING)]
назвать буквы1 [вершина]
назвать буквы [действие]
1 (тип: STRING) [ожидаемое значение (сорт: STRING)]
1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
назвать буквы2 [вершина]
назвать буквы [действие]
2 (тип: STRING) [ожидаемое значение (сорт: STRING)]
1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[вершина]

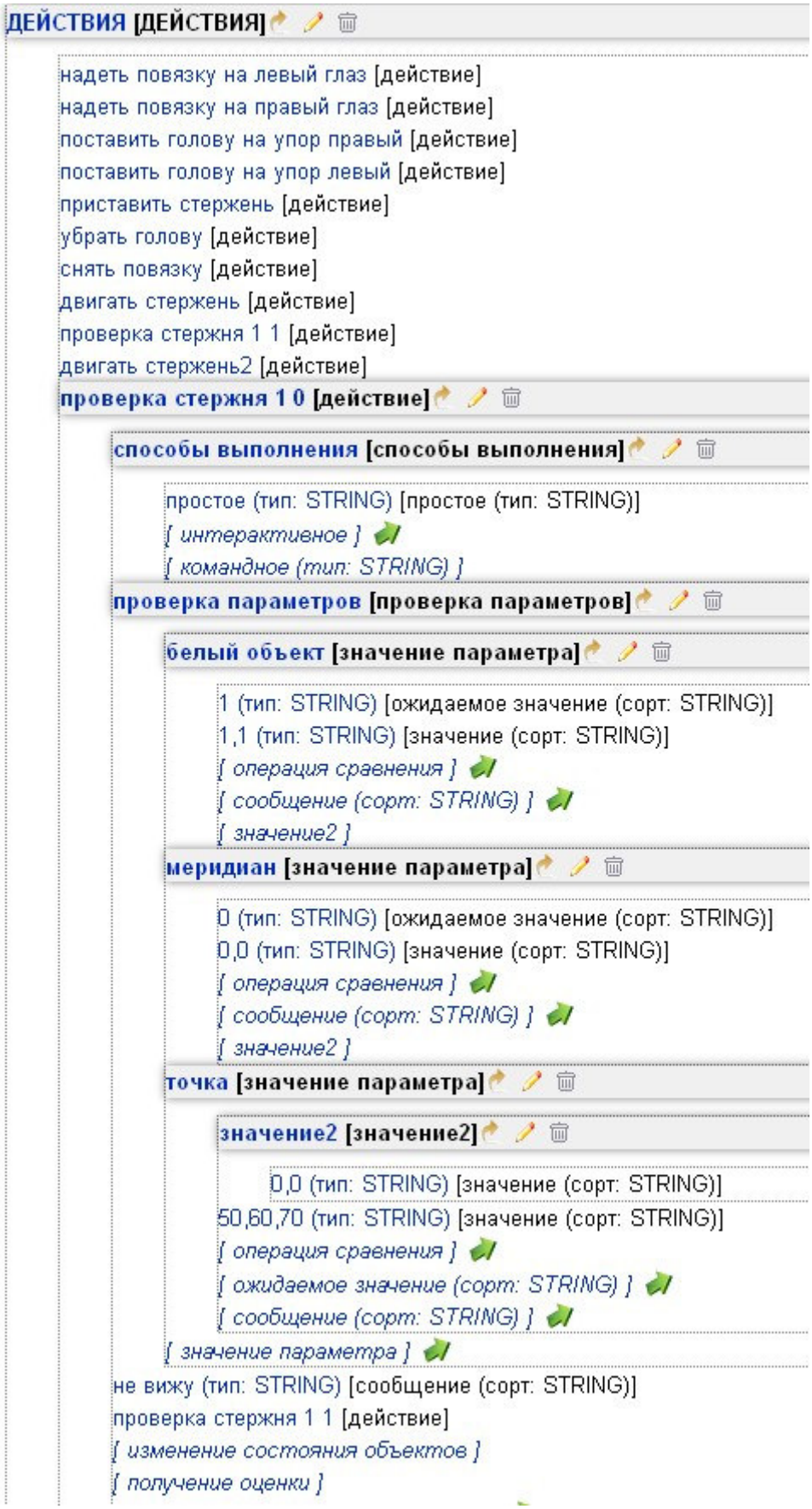
Реализация декларативной модели компьютерного тренажера для обучения проверке зрения по таблицам Сивцева в структурном редакторе

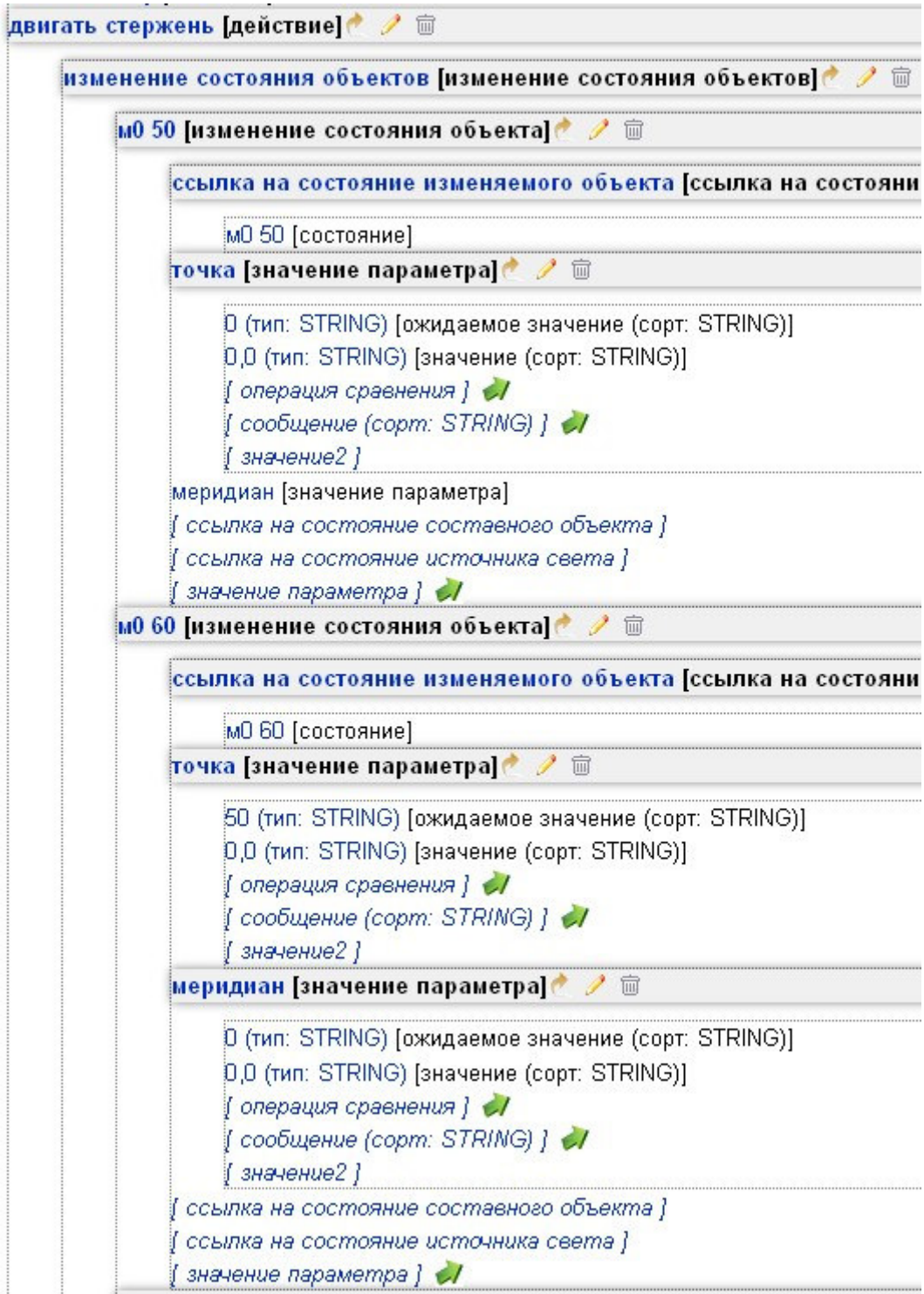


пациент [изменяемый объект]
- логические атрибуты [логические атрибуты]
  - 1мм м0 [атрибут]
    - 50,60,70 (тип: STRING) [значение (сорт: STRING)]
  - 1мм м1 [атрибут]
    - 50,60,70 (тип: STRING) [значение (сорт: STRING)]
  - *[атрибут]*
- презентационные атрибуты [презентационные атрибуты]
- *[множество состояний]*
- *[состояние]*

кампиметр [простой объект]
- презентационные атрибуты [презентационные атрибуты]
- *[логические атрибуты]*

стержень [изменяемый объект]
- множество состояний [множество состояний]
  - пустой [состояние]
  - объект 1мм [состояние]
  - объект 3мм [состояние]
  - объект 5мм [состояние]
  - объект 10мм [состояние]
  - м0 0 [состояние]
  - м0 50 [состояние]
  - м0 60 [состояние]
  - м0 70 [состояние]
  - м1 0 [состояние]
  - м1 50 [состояние]
  - м1 60 [состояние]
  - м1 70 [состояние]
  - *[состояние]*
- логические атрибуты [логические атрибуты]
  - точка [атрибут]
    - 0,0 (тип: STRING) [значение (сорт: STRING)]
  - белый объект [атрибут]
    - 1,1 (тип: STRING) [значение (сорт: STRING)]
  - меридиан [атрибут]
    - 0,0 (тип: STRING) [значение (сорт: STRING)]
  - *[атрибут]*
- презентационные атрибуты [презентационные атрибуты]
- м0 0 [состояние]


ДЕЙСТВИЯ [ДЕЙСТВИЯ]
надеть повязку на левый глаз [действие]
надеть повязку на правый глаз [действие]
поставить голову на упор правый [действие]
поставить голову на упор левый [действие]
приставить стержень [действие]
убрать голову [действие]
снять повязку [действие]
двигать стержень [действие]
проверка стержня 1 1 [действие]
двигать стержень2 [действие]
проверка стержня 1 0 [действие]
способы выполнения [способы выполнения]
простое (тип: STRING) [простое (тип: STRING)]
[интерактивное]
[командное (тип: STRING)]
проверка параметров [проверка параметров]
белый объект [значение параметра]
1 (тип: STRING) [ожидаемое значение (сорт: STRING)]
1,1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
меридиан [значение параметра]
0 (тип: STRING) [ожидаемое значение (сорт: STRING)]
0,0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
точка [значение параметра]
значение2 [значение2]
0,0 (тип: STRING) [значение (сорт: STRING)]
50,60,70 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[ожидаемое значение (сорт: STRING)]
[сообщение (сорт: STRING)]
[значение параметра]
не вижу (тип: STRING) [сообщение (сорт: STRING)]
проверка стержня 1 1 [действие]
[изменение состояния объектов]
[получение оценки]


двигать стержень [действие]
изменение состояния объектов [изменение состояния объектов]
м0 50 [изменение состояния объекта]
ссылка на состояние изменяемого объекта [ссылка на состояни
м0 50 [состояние]
точка [значение параметра]
0 (тип: STRING) [ожидаемое значение (сорт: STRING)]
0,0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
меридиан [значение параметра]
[ссылка на состояние составного объекта]
[ссылка на состояние источника света]
[значение параметра]
м0 60 [изменение состояния объекта]
ссылка на состояние изменяемого объекта [ссылка на состояни
м0 60 [состояние]
точка [значение параметра]
50 (тип: STRING) [ожидаемое значение (сорт: STRING)]
0,0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
меридиан [значение параметра]
0 (тип: STRING) [ожидаемое значение (сорт: STRING)]
0,0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
[ссылка на состояние составного объекта]
[ссылка на состояние источника света]
[значение параметра]

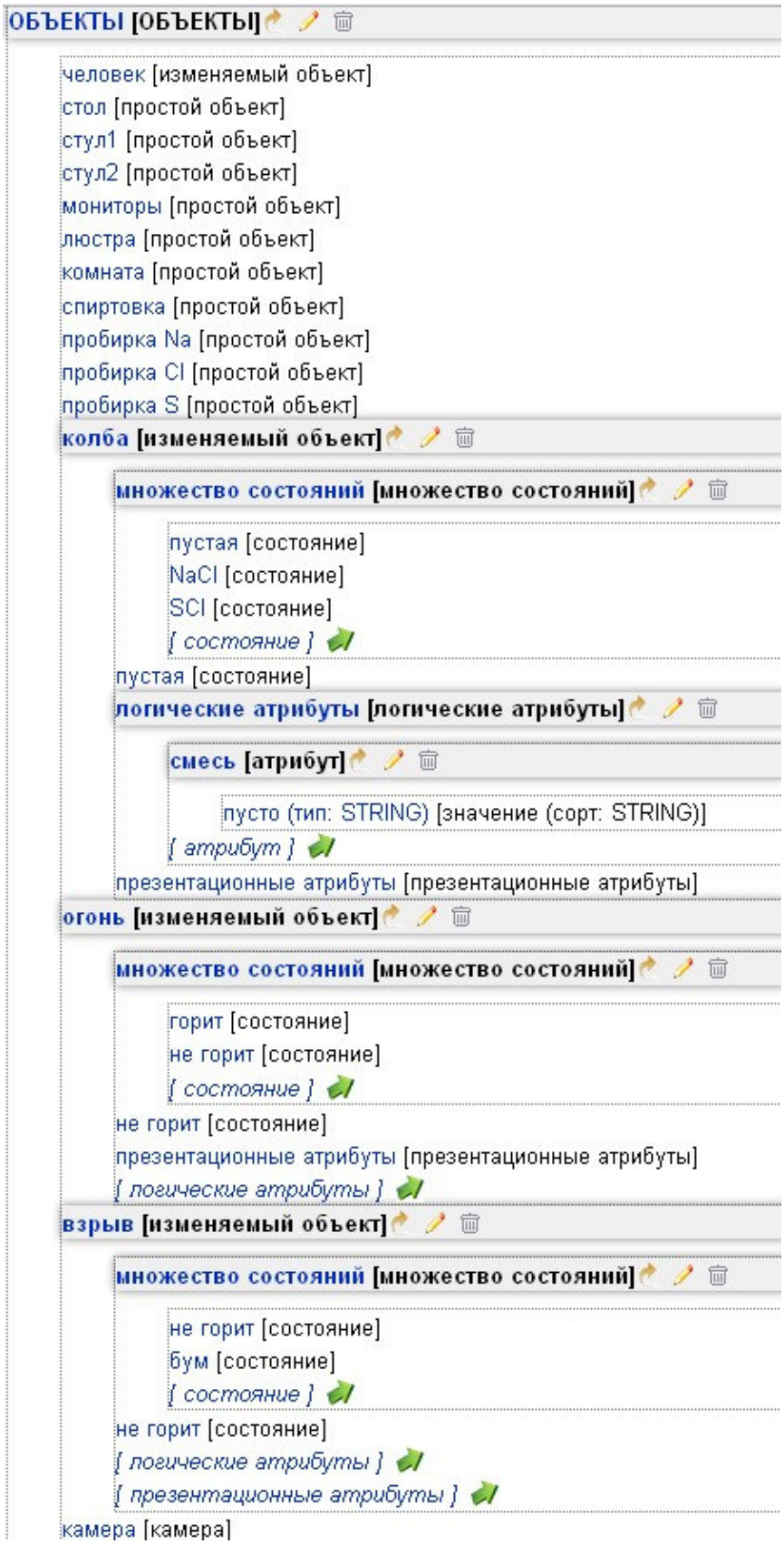

м0 70 [изменение состояния объекта]
ссылка на состояние изменяемого объекта [ссылка на сос
м0 70 [состояние]
точка [значение параметра]
60 (тип: STRING) [ожидаемое значение (сорт: STRING)]
0,0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
меридиан [значение параметра]
0 (тип: STRING) [ожидаемое значение (сорт: STRING)]
0,0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
[ссылка на состояние составного объекта]
[ссылка на состояние источника света]
[значение параметра]
м1 0 [изменение состояния объекта]
ссылка на состояние изменяемого объекта [ссылка на сос
м1 0 [состояние]
точка [значение параметра]
70 (тип: STRING) [ожидаемое значение (сорт: STRING)]
0,0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
меридиан [значение параметра]
0 (тип: STRING) [ожидаемое значение (сорт: STRING)]
0,0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
[ссылка на состояние составного объекта]
[ссылка на состояние источника света]
[значение параметра]


м1 50 [изменение состояния объекта]
ссылка на состояние изменяемого объекта [ссылка на состоян
м1 50 [состояние]
точка [значение параметра]
0 (тип: STRING) [ожидаемое значение (сорт: STRING)]
0,0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
меридиан [значение параметра]
1 (тип: STRING) [ожидаемое значение (сорт: STRING)]
0,0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
[ссылка на состояние составного объекта]
[ссылка на состояние источника света]
[значение параметра]
м1 60 [изменение состояния объекта]
ссылка на состояние изменяемого объекта [ссылка на состоян
м1 60 [состояние]
точка [значение параметра]
50 (тип: STRING) [ожидаемое значение (сорт: STRING)]
0,0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
меридиан [значение параметра]
1 (тип: STRING) [ожидаемое значение (сорт: STRING)]
0,0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
[ссылка на состояние составного объекта]
[ссылка на состояние источника света]
[значение параметра]


двигать стержень [действие]
изменение состояния объектов [изменение состояния объектов]
м0 50 [изменение состояния объекта]
м0 60 [изменение состояния объекта]
м0 70 [изменение состояния объекта]
м1 0 [изменение состояния объекта]
м1 50 [изменение состояния объекта]
м1 60 [изменение состояния объекта]
м1 70 [изменение состояния объекта]
ссылка на состояние изменяемого объекта [ссылка на состояни
м1 70 [состояние]
точка [значение параметра]
60 (тип: STRING) [ожидаемое значение (сорт: STRING)]
0,0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
меридиан [значение параметра]
1 (тип: STRING) [ожидаемое значение (сорт: STRING)]
0,0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
[ссылка на состояние составного объекта]
[ссылка на состояние источника света]
[значение параметра]
[изменение состояния объекта]
способы выполнения [способы выполнения]
по кампиметру [интерактивное]
событие [событие]
MOUSECLICK (тип: STRING) [MOUSECLICK (тип: STRING)]
кампиметр [простой объект]
[таблица]
[изменяемый объект]
[составной объект]
[ссылка на состояние изменяемого объекта]
[ссылка на состояние составного объекта]
[интерактивное]
[простое (тип: STRING)]
[командное (тип: STRING)]


проверка стержня 1 1 [действие]
проверка параметров [проверка параметров]
белый объект [значение параметра]
1 (тип: STRING) [ожидаемое значение (сорт: STRING)]
1,1 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
меридиан [значение параметра]
1 (тип: STRING) [ожидаемое значение (сорт: STRING)]
0,0 (тип: STRING) [значение (сорт: STRING)]
[операция сравнения]
[сообщение (сорт: STRING)]
[значение2]
точка [значение параметра]
0,0 (тип: STRING) [значение (сорт: STRING)]
значение2 [значение2]
[операция сравнения]
[ожидаемое значение (сорт: STRING)]
[сообщение (сорт: STRING)]
[значение параметра]
способы выполнения [способы выполнения]
простое (тип: STRING) [простое (тип: STRING)]
[интерактивное]
[командное (тип: STRING)]
не вижу (тип: STRING) [сообщение (сорт: STRING)]
[изменение состояния объектов]
[получение оценки]
[параметры обработки агентами]

Реализация декларативной модели виртуальной химической лаборатории




ОБЪЕКТЫ [ОБЪЕКТЫ]
ДЕЙСТВИЯ [ДЕЙСТВИЯ]
очистить колбу [действие]
способы выполнения [способы выполнения]
командное (тип: STRING) [командное (тип: STRING)]
[интерактивное]
[простое (тип: STRING)]
изменение состояния объектов [изменение состояния объектов]
очищение колбы [изменение состояния объекта]
[изменение состояния объекта]
параметры обработки агентами [параметры обработки агентами]
смесь [параметр обработки агентом]
[параметр обработки агентом]
[сообщение (сорт: STRING)]
[получение оценки]
[действие]
[проверка параметров]
[звук (сорт: STRING)]
смешать [действие]
способы выполнения [способы выполнения]
командное (тип: STRING) [командное (тип: STRING)]
[интерактивное]
[простое (тип: STRING)]
изменение состояния объектов [изменение состояния объектов]
в колбе NaCl [изменение состояния объекта]
в колбе SCl [изменение состояния объекта]
в колбе NaS огонь [изменение состояния объекта]
[изменение состояния объекта]
[сообщение (сорт: STRING)]
[получение оценки]
[параметры обработки агентами]
[действие]
[проверка параметров]
[звук (сорт: STRING)]


нагреть [действие]
способы выполнения [способы выполнения]
командное (тип: STRING) [командное (тип: STRING)]
[интерактивное]
[простое (тип: STRING)]
изменение состояния объектов [изменение состояния объектов]
нагрев спиртовки [изменение состояния объекта]
[изменение состояния объекта]
-> смешать [действие]
параметры обработки агентами [параметры обработки агентами]
смесь [параметр обработки агентом]
[параметр обработки агентом]
[сообщение (сорт: STRING)]
[получение оценки]
[проверка параметров]
[звук (сорт: STRING)]
добавить натрий [действие]
способы выполнения [способы выполнения]
параметры обработки агентами [параметры обработки агентами]
-> смешать [действие]
[сообщение (сорт: STRING)]
[изменение состояния объектов]
[получение оценки]
[проверка параметров]
[звук (сорт: STRING)]


добавить натрий [действие]
способы выполнения [способы выполнения]
параметры обработки агентами [параметры обработки агентами]
-> смешать [действие]
[сообщение (сорт: STRING)]
[изменение состояния объектов]
[получение оценки]
[проверка параметров]
[звук (сорт: STRING)]
добавить хлор [действие]
способы выполнения [способы выполнения]
командное (тип: STRING) [командное (тип: STRING)]
[интерактивное]
[простое (тип: STRING)]
параметры обработки агентами [параметры обработки агентами]
смесь [параметр обработки агентом]
[параметр обработки агентом]
-> смешать [действие]
[сообщение (сорт: STRING)]
[изменение состояния объектов]
[получение оценки]
[проверка параметров]
[звук (сорт: STRING)]
добавить серу [действие]
способы выполнения [способы выполнения]
командное (тип: STRING) [командное (тип: STRING)]
[интерактивное]
[простое (тип: STRING)]
параметры обработки агентами [параметры обработки агентами]
смесь [параметр обработки агентом]
[параметр обработки агентом]
-> смешать [действие]
[сообщение (сорт: STRING)]
[изменение состояния объектов]
[получение оценки]
[проверка параметров]
[звук (сорт: STRING)]

ПРИЛОЖЕНИЕ 4

"Согласовано"
Проректор по научной работе
В.А. Невзорова

" Утверждаю"
Проректор по учебной работе
Е.В. Крукович

АКТ

о внедрении научных исследований
в учебный процесс

Мы, представители кафедры офтальмологии и оториноларингологии: зав. кафедрой офтальмологии и оториноларингологии ТГМУ, заслуженный врач РФ, д.м.н., профессор В.Я. Мельников; завуч кафедры офтальмологии и оториноларингологии ТГМУ, главный офтальмолог департамента здравоохранения Администрации Приморского края и ДФО, заслуженный врач РФ, к.м.н., доцент Л.П. Догадова; к.м.н. Н.В. Филина, настоящим актом подтверждаем, что:

"Обучающий компьютерный тренажер по классическим методам исследования в офтальмологии", разработанный в лаборатории интеллектуальных систем Федерального государственного бюджетного учреждения науки Института автоматики и процессов управления ДВО РАН (разработчик Федорищев Леонид Александрович), внедрен на кафедре 31.10.2013
(указать дату внедрения)

Форма внедрения:

- компьютерная программа: "Обучающий компьютерный тренажер по классическим методам исследования в офтальмологии", для курса офтальмологии, лечебного факультета.

Эффективность внедрения:

- повышение качества подготовки: у студентов лечебного факультета по курсу "офтальмология" и дистанционного обучения врачей ФУВ.

Подпись - зав. кафедрой Мельников В.Я.

Дата: 31.10.2013