* * *

Использование встроенных таблиц интерпретатора совместно с интеллектуальным агентом, снабженным соответствующим механизмом связей дескрипторов и функций в этой схеме программного обеспечения, позволяет программному обеспечению модифицироваться самостоятельно или при помощи простого пользователя без вмешательства специалистов.

Данная технология была успешно применена для анализа гистологических изображений. В дальнейшем она будет адаптироваться к другим задачам, связанным с обработкой изображений. Развитие данной технологии будет происходить по сценарию: от построения концепции до формирования интерфейса пользователя и алгоритма решения задачи.

В настоящее время представленная технология реализована в экспериментальном программном комплексе QTIP, который использовался в качестве дополнительных средств для диагностики заболевания по гистологическим изображениям. В дальнейшем комплекс будет адаптироваться для наноскопических изображений и изображений спутниковой или аэросъемки земной поверхности. Для достижения этой цели необходимо сформировать новые дескриптивные признаки, которые будут связаны с функциями обработки изображений. Кроме того, общая концепция построения такого программного обеспечения, безусловно, проявит себя в ближайшей перспективе при построении сложных программных комплексов.

1. Гулиева И. Ф., Рюмина Е. В., Гулиев Я. И. // Врач и информ. технологии. 2009. № 3. С. 4.
2. Абламейко С. В., Недзьведь А. М. Обработка оптических изображений клеточных структур в медицине. Минск, 2005.
3. Marceau G. The speed, size and dependability of programming languages // Blog Square root of x divided by zero, Saturday, May 30, 2009. Mode of access: http://gmarceau.qc.ca/blog/2009/05/speed-size-and-dependability-of.html.
4. Reitmayr G., Schmalstieg D. // Proc. of VRST'01, November 15–17, 2001. Banff, Alberta. P. 47.
5. Цыбульник В. А. // Рос. журн. программистов RSDN. 2010. № 2. С. 39.
6. Kaski S. // Acta Polytechnica Scandinavica, Mathematics. Computing and Management in Engineering Series. Espoo, 1997. № 82.
7. Васильев И. А. // Искусств. интеллект. 2003. № 3. С. 48.
8. Ierusalimschy R. Programming in Lua 2nd ed. Lua. org. 2006.
9. Bradski G., Kaehler A. Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly, 2008.

Поступила в редакцию 27.08.12.

Александр Михайлович Недзьведь – кандидат технических наук, ведущий научный сотрудник Объединенного института проблем информатики НАН Беларуси.

Сергей Владимирович Абламейко – академик НАН Беларуси, доктор технических наук, профессор, ректор БГУ.

УДК 681.3.06+347.78.031

Г. А. ЛОМАКИН, А. С. ГУСЕЙНОВА

ОБЩИЕ ПОДХОДЫ К РАЗРАБОТКЕ ИНТЕГРИРОВАННОЙ СРЕДЫ ДЛЯ ПОСТРОЕНИЯ ГРАФИЧЕСКИХ ПРИЛОЖЕНИЙ

The paper describes the main approaches for creating interactive web environment for building 3D-applications. The modern development of graphics systems and technologies is urgent transferring of imaging to the next level, which greatly simplifies the development of 3D-objects. Scientific novelty of the approach is to create an integrated system for the implementation of graphics applications that abstracts the end user from the intermediate levels graphics creation and provides the ability to directly construct graphic scenes through a graphical interface on modern platforms such as Windows 7 and Android. The main concept of the proposed solution involves the creation of a set of classes and utilities that implement a common approach to graphics rendering and hide by the direct use of various mathematical models and graphics algorithms. This will allow a wider range of stakeholders to create the desired 3D-objects and scenes, spending minimal time to develop the necessary solutions.

Ключевые слова: 3D-приложение, графическое ядро, фреймворк, контент, Android, графический интерфейс пользователя.
Key words: 3D-application, graphic kernel, framework, content, Android, graphic interface of the user.

Компьютерная графика достаточно широко используется для решения задач визуализации как объектов реального мира, так и несуществующих изображений различных объектов. Значительный научный и практический интерес представляют также задачи создания виртуальных реальностей, которые отождествляют существующие прототипы, например, визуализации различных памятников архитектуры и археологии. Отметим также, что при современном развитии графических систем и технологий актуален подход, связанный с переводом визуализации 3D-объектов на новый уровень, позволяющий упростить разработку 3D-графики. Таким образом, реализация обобщенной среды (графического ядра) для визуализации различных объектов, которая предоставит пользователю возможность быстрого создания 3D-объектов, используя современные интернет- и мобильные технологии, является востребованным направлением в развитии компьютерной графики.

Концепция, лежащая в основе предлагаемого решения, связана с набором классов и утилит, которые реализуют более общие подходы к визуализации графических объектов и скрывают от пользователя непосредственное использование тех или иных математических моделей и графических алгоритмов. Все это позволит более широкому кругу заинтересованных лиц создавать требуемые 3D-объекты и сцены, затрачивая минимальное время на разработку необходимого решения.

Общая характеристика реализуемой среды. Следует отметить, что на данный момент существует небольшое количество доступных сред для создания интерактивных 3D-визуализаций, к недостаткам которых можно отнести отсутствие хранилища различного 3D-контента в облаке и возможностей, предоставляющих пользователю высокоуровневый подход для работы с 3D-пространством и различными объектами.

Поэтому важно сформулировать требования, связанные с разработкой среды. Прежде всего это:

- определение конкретных рамок и цели создаваемого графического ядра;
- обеспечение доступа к контенту;
- реализация набора программ-утилит для использования контента и графического ядра под различные платформы;
- реализация фреймворка с высоким уровнем абстракций;
- обеспечение широкого выбора различных алгоритмов для рендеринга изображения.

Предлагаемая интерактивная среда состоит из нескольких компонент:

- облака для хранения контента в различных категориях;
- графического ядра, реализованного с использованием OpenGL и DirectX;
- набора утилит для персонального компьютера;
- приложения для мобильной операционной системы Android;
- фреймворка, обеспечивающего доступ ко всем возможностям ядра.

Таким образом, главными особенностями предлагаемой веб-среды являются:

- открытое ядро на OpenGL и DirectX;
- набор классов и интерфейсов для рендеринга примитивов, а также набор базовых шейдеров для реализации различных эффектов на графическом конвейере;
- утилиты для построения визуализаций на платформах PC и Android;
- синхронизация контента клиентского приложения с сервером.

Рассмотрим аспекты, связанные с основными архитектурными и конструктивными особенностями предлагаемой интегрированной среды разработки 3D-приложений.

Архитектура интегрированной веб-среды. Предлагаемая веб-среда для построения графических приложений абстрагирует пользователя от промежуточных уровней создания графики и дает возможность напрямую реализовывать графические сцены посредством GUI, в том числе обеспечивает пользователя шаблонным контентом [1].

Для достижения поставленной задачи были выполнены следующие модули:

- ядро для построения сложных графических сцен, предоставляющее открытое API [2];
- графическое ядро на OpenGL для ОС Windows 7 и OpenGL ES 2.0 для ОС Android [3];
- набор утилит, использующих ядро для манипулирования графическими процессами с помощью GUI [4];
- сервер с контентом (доступ к серверу также осуществляется напрямую из утилит) для предоставления возможности использования шаблонного наполнения, а также загрузки произвольного контента.

В общем виде архитектура предлагаемой веб-среды представлена на рис. 1.

Реализация графического ядра CoreX. В качестве оптимального решения для визуализации сложных трехмерных объектов реального и виртуального мира предлагается графическое ядро CoreX, базирующееся на использовании XNA-фреймворка, который включает Content Pipeline. Это позволяет упростить работу с контентом и при необходимости расширить список поддерживаемых типов.

Графические объекты представляются в качестве классов. Класс каждого графического объекта описывает его основные свойства, а также методы для отображения объекта в пространстве и обновления его состояния. Каждый класс связывается с конкретным контентом, в нем же определяется, как данный контент будет использоваться.

На базе ядра CoreX реализован также и графический интерфейс пользователя (GUI), позволяющий производить интеграцию различных элементов управления в сцене и манипулировать происходящим на ней. Кроме графической составляющей в CoreX присутствует и звук.

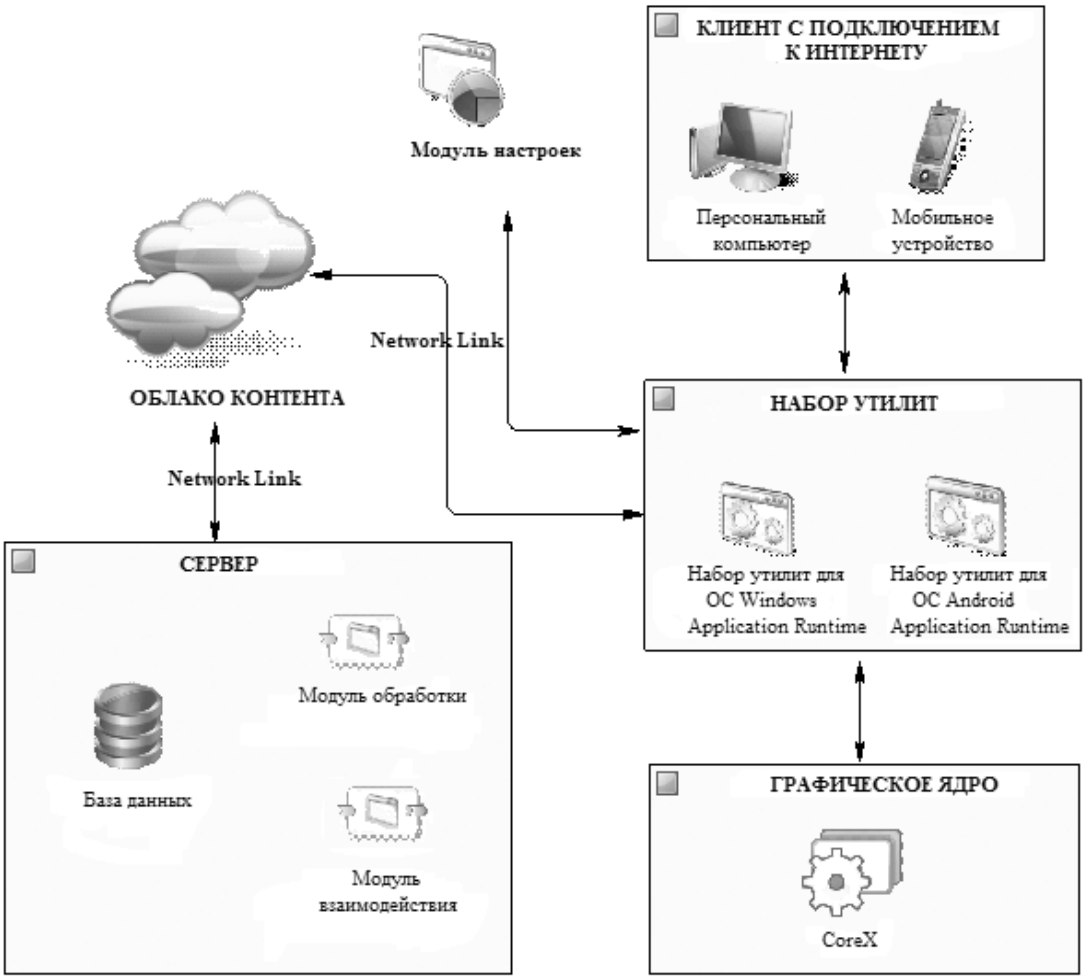

Рис. 1. Архитектура интегрированной веб-среды для построения графических приложений

Графическое ядро CoreX поддерживает DirectX 10, а также шейдерную модель версии 4.0. Реализованы такие эффекты, как per-Pixel lighting, bump mapping, parallax mapping, shadow mapping, shadow volumes, multisampling.

Созданное ядро предоставляет API для построения приложений. Разработанное API позволяет строить и изменять сцену, добавлять новые объекты для рендеринга и изменять их параметры (положение в пространстве, коэффициенты масштабирования, углы наклона относительно осей, текстуры, шейдер и др.). Под объектами подразумеваются различные 3D-meshes. Их загрузка в ядро также осуществляется с помощью API. Кроме этого, есть возможность применения глобальных эффектов ко всей сцене (таких как глобальное освещение/затенение, motion blur, HDR и др.).

Интерфейс взаимодействия пользователя с ядром. Как уже говорилось, пользователь взаимодействует с ядром при помощи универсальной утилиты, состоящей из двух основных частей. Первая часть отвечает за контент и объекты, использующие этот контент, а также за свойства и различные характеристики этих объектов, вторая – за настройку ядра CoreX – детализацию, качество текстур, используемые технологии и др.

После конфигурации все данные передаются на ядро, после чего запускается визуализация.

Следует отметить, что при задании параметров объектов имеется также возможность предварительного просмотра и взаимодействия с объектами и их свойствами посредством работы в окне предварительного просмотра. Таким образом, пользователь всегда видит, где размещает объект и как он расположен относительно других объектов.

Создание сцены происходит следующим образом. Пользователь выбирает понравившийся ему контент, затем перетягивает его из окна диалога в Viewport, настраивает параметры контента и вписывает в общую композицию. По желанию пользователь может применить ко всей сцене глобальные эффекты в отдельном диалоге. Для достижения динамических сцен и эффектов используются скрипты, которые можно применить к различным объектам, событиям и к циклу рендеринга в целом.

Для их написания используется «облегченный» синтаксис Java Script. При помощи скриптов можно динамически изменять значения различных параметров объектов, например положение в пространстве, габариты, режимы освещения и текстурирования, и в итоге запустить сцену в режиме исполнения, в результате чего создается файл.

На рис. 2 изображен диалог изменения свойств объекта. Взаимодействуя с ним, можно корректировать различные параметры объекта, такие как положение в пространстве, углы поворота, текстура, эффекты и т. д.

Рис. 2. Диалог изменения свойств объекта

Серверная часть. Для предоставления пользователю шаблонных ресурсов разработано дистанционное хранилище контента. Пользователю также доступна возможность загрузки на сервер ресурсов для общего использования. Доступ к серверу осуществляется как из утилит, так и с помощью веб-интерфейса, включая мобильные платформы.

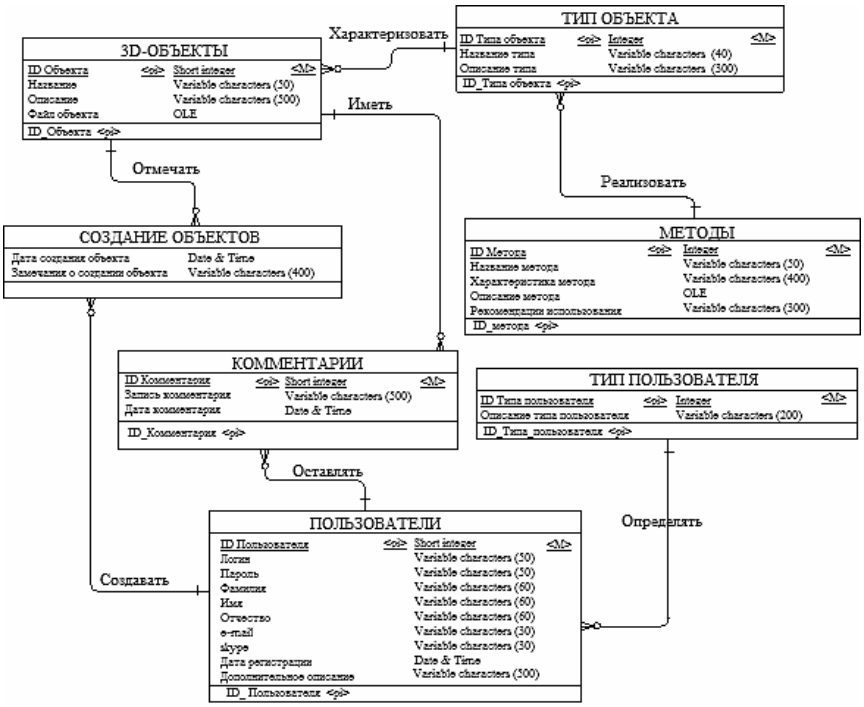

Рис. 3. Фрагмент концептуальной модели базы данных, предназначенный для создания и хранения 3D-объектов

На рис. 3 представлен фрагмент модели базы данных, предназначенный для создания и хранения ряда основных сущностей, среди которых 3D-объекты, Тип объекта, Методы, Создание объектов, Пользователи, Тип пользователя, Комментарии. Каждый объект, который помещается в базу, характеризуется типом и методом его реализации. У пользователя есть возможность добавлять комментарии по поводу методов реализации, характеристик объекта в процессе использования. В зависимости от типа пользователя назначаются соответствующие привилегии по работе с 3D-объектами. Отметим также, что в модели данных поддерживается декларативная ссылочная целостность.

Таким образом, приложение, построенное на базе графического ядра, представляет собой универсальную утилиту для создания сцены, придания различным объектам требуемых физических свойств, а также задания необходимых взаимодействий объектов. Все это осуществляется благодаря работе с графическими объектами и контентом посредством графического интерфейса и изменения свойств объектов. После построения сцены осуществляется переход в режим наблюдения, в котором можно видеть все заданные эффекты, движения, столкновения, освещения и т. д., а также самому взаимодействовать с созданной средой.

Пространство имен разработанного CoreX можно применять для реализации собственных приложений как графической компоненты, например, при создании компьютерных игр. В данном случае пользователю требуется лишь определиться с контентом и задать игровую логику.

Отметим, что приложения, созданные с использованием этого ядра, являются кроссплатформенными. В данный момент поддерживается Windows 7 и Android. В дальнейшем появится поддержка Windows Phone 7. Более того, так как CoreX является пространством имен в .Net, то можно использовать его классы также и в любом .Net-приложении.

1. Мэрдок Келли Л. Autodesk 3ds Max 2009. Библия пользователя. 3D Studio max. М., 2008. С. 656.
2. Ломакин Г. А. // Материалы Международного конгресса по информатике: информационные системы и технологии (CSIST'2011), Минск, 1 окт. – 3 нояб. 2011 г.: в 2 ч. Минск, 2011. Ч. 2. С. 367.
3. Ломакин Г. А. // Веб-программирование и интернет-технологии WebConf2012: материалы 2-й Междунар. науч.-практ. конф., Минск, 5–7 июня 2012 г. Минск, 2012. С. 164.
4. Ломакин Г. А. // Сборник работ 68-й Научной конференции студентов и аспирантов Белорусского государственного университета, Минск, 1–20 мая 2011 г.: в 3 ч. Минск, 2011. Ч. 1. С. 124.

Поступила в редакцию 30.06.12.

Герман Александрович Ломакин – студент 4-го курса факультета прикладной математики и информатики.
Анастасия Сергеевна Гусейнова – ассистент кафедры многопроцессорных систем и сетей.

УДК 004.045

А. П. ПОБЕГАЙЛО

РЕАЛИЗАЦИЯ ШАБЛОНА ПРОЕКТИРОВАНИЯ *ACTOR – ROLE*, ОСНОВАННАЯ НА ИДИОМАХ ПРЕОБРАЗОВАНИЯ ТИПОВ ДАННЫХ

An implementation of *Actor – Role* design pattern by means of languages with static types is considered. The proposed approach is based on a realization of the association between roles and actors by means of the association class. The implementation of the association class is based on idioms of type promotion. The programming language C++ is used as a model language for the proposed implementation of *Actor – Role* design pattern. The source code of two frameworks for realization of role assignment and role choosing is presented. Dynamic type identification mechanism of the programming language C++ is used to determine which role is chosen for the actor.

Ключевые слова: шаблоны проектирования, шаблон *Актер – Роль*, назначение ролей, выбор роли, преобразование типов данных, язык программирования C++.
Key words: design patterns, *Actor – Role* design pattern, role assignment, choosing roles, type promotion, C++ programming language.

В настоящее время этап проектирования программных систем включает разработку концептуальной модели прикладной области. Каждая концепция рассматривается как некоторая сущность, обладающая определенными свойствами и связями с другими концепциями. Поэтому естественно, что в концептуальной модели прикладной системы встречаются такие сущности, как роли. Например, человек может играть роль врача, учителя и так далее в зависимости от ситуации. Так как концептуальная модель прикладной области естественным образом является основой концептуальной модели программной системы, то при разработке таких систем требуется решить вопрос о реализации ролей или их назначении объектам классов. Более подробно вопросы концептуального моделирования прикладной области системы с использованием ролей рассмотрены в работах [1–5].