УДК 004.94, 004.85

Е. Л. Калишенко, Ю. В. Романенко

Технологические аспекты реализации компьютерных обучающих систем с трехмерной графикой в среде QNX

Описываются технологические аспекты разработки трехмерных графических приложений в составе компьютерных обучающих систем. Рассматриваются различные варианты оптимизации и повышения производительности работы графических приложений в среде QNX.

Компьютерное обучение, тренажеры, трехмерные графические приложения, операционная система QNX, оптимизация работы графических приложений

В современных экономических условиях, когда даже мелкий ремонт сложного технологического оборудования сложных технических комплексов грозит превратиться в неразрешимую проблему, цена ошибок эксплуатирующего персонала многократно возрастает. Особую важность приобретает качественное обучение и постоянное поддержание квалификации и готовности персонала, его противоаварийные тренировки. Актуальной является проблема создания адекватной системы обучения и аттестации, способной быстро реагировать на изменяющиеся требования и условия эксплуатации сложных систем, устройств, механизмов.

В последние годы в практике инженерной подготовки все чаще используют компьютерные обучающие системы (КОС). КОС представляют собой аппаратно-программные комплексы, которые легко перенастраиваются при модернизации оборудования, позволяют построить обучение на основе проблемно-ситуационного подхода, реализовать активные (интерактивные) методики подготовки. КОС создают возможность реализовать практически любые по сложности эксперименты с оборудованием и воспроизвести методики отработки любых штатных и нештатных ситуаций, не используя реальные устройства и агрегаты.

Современные КОС по назначению и функциональным возможностям условно разделяются на следующие группы:

– *компьютерные учебные пособия* – программно-методические комплексы, обеспечивающие возможность самостоятельно освоить учебный курс, соединяющие в себе свойства обычного учебника, справочника, задачника. Обычно компьютерные учебные пособия выполняются в виде электронных презентаций, учебных фильмов и интерактивных гипертекстовых баз знаний и т. д.;

– *интерактивные обучающие комплексы* – это пакеты программ, позволяющие оператору самостоятельно изучить учебный материал в интерактивном режиме и провести самопроверку;

– *компьютерные тренажеры* – аппаратно-программные комплексы, реализующие формализованную модель изменения состояния систем управления техническими комплексами в зависимости от действий операторов и задаваемых внешних факторов. Обучаемый взаимодействует с виртуальными устройствами, внешний вид и технология работы с которыми повторяют реальное оборудование. Такие тренажеры можно использовать для индивидуального и группового обучения, моделирования штатных технологических процессов и нештатных ситуаций, отработки и закрепления практических навыков совместной работы с оборудованием. Они обеспечивают возможность обучения и тренировки на различных уровнях самостоятельности обучаемого, контроль действий операторов.

Одной из важнейших функций КОС, определяющей качество и результативность обучения эксплуатирующего персонала, является отображение объектов реального мира. Реализация этой функции связана с воспроизведением визуальных образов с использованием мультимедиа-технологий, многовариантной системы моделирования, имитацией управления объектами путем интерактивного взаимодействия. Наглядность представления и удобство работы оператора с КОС дости-

гаются за счет современных компьютерных технологий, включая видеоизображения, интерактивные двухмерные и трехмерные модели с возможностью изменения параметров отображения и обзора модели, анимационные ролики. Визуальные образы объектов в большинстве современных КОС создаются с помощью программно-генерируемой графики. Для реализации движений, динамических процессов или физических явлений используется программная анимация. Максимальный эффект обучения достигается за счет использования трехмерных графических приложений.

В общем понимании трехмерные графические приложения представляют собой специальное программное обеспечение, позволяющее просматривать трехмерные визуальные образы объектов реального мира. При этом при отображении могут преследоваться следующие цели:

– отображение внешнего вида объектов;

– отображение состояния реальных объектов в виде трехмерных образов.

В первом случае графическое приложение довольно простое, так как создается статическое изображение объекта. Объекты имеют фиксированное состояние (внешний вид) и расположение. Оператор не имеет возможности управлять отображением, изменять или просматривать состояние объекта, получать дополнительную информацию по объектам. Единственное, что может делать оператор, это изменять точку обзора (положение камеры). В простейшем случае приложение может быть сведено к некоторой последовательности изображений (видеофильму). Интерактивность достигается за счет управление воспроизведением при помощи клавиатуры или мыши. Для создания подобных приложений обычно используются специализированные редакторы (редакторы сцен), позволяющие импортировать данные из трехмерных моделей и размещать их на сцене. Написания специального программного кода не требуется. Процесс создания таких приложений состоит из следующих этапов:

1. Разработка моделей в графическом редакторе.

2. Экспорт моделей в редактор сцены, размещение моделей на сцене.

3. Экспорт сцены в проигрыватель.

Если проигрыватель поддерживает непосредственный импорт моделей, созданных в графическом редакторе, то этап создания сцены можно пропустить.

Приложения, отображающие состояния реальных объектов в виде трехмерных образов, имеют более сложную структуру. Помимо отображения трехмерных моделей они позволяют отображать состояние отдельных объектов сцены и их изменения. Разработка таких приложений практически невозможна без написания специального программного кода, так как необходимо задавать связи объектов с некоторыми структурами данных, хранящими их состояние. Данные о состоянии могут передаваться в приложение из базы данных, файла или по сетевому протоколу. Возможен вариант использования некоторого графического редактора (аналог редактора сцен), который позволит не только формировать сцену, но и задавать связи объектов с данными.

В общем случае разработка трехмерных графических приложений под QNX состоит из следующих этапов:

1. Определение требований к графическому приложению. На данном этапе определяется необходимый функционал, который должно реализовывать приложение, требования по производительности, поддерживаемым операционным системам и т. д.

2. Сбор исходных данных для разработки приложений. На данном этапе определяется состав оборудования, отображаемого в графическом приложении, подготавливаются чертежи, схемы, фотографии, технические описания, которые позволят разработать трехмерные модели оборудования, отображаемого в приложении. При этом, так как процесс моделирования автоматизирован, для ускорения процесса разработки чертежи могут быть представлены в цифровом виде.

3. Разработка трехмерных моделей оборудования. На данном этапе в специализированных программах трехмерного моделирования на основе исходных данных, полученных на предыдущем этапе, создаются визуальные образы оборудования. Для данных образов создаются текстуры, настраиваются материалы и т. д. На выходе получаются модели в формате среды разработки.

4. Выбор графического «движка», на основе которого будет разрабатываться приложение.

5. Экспорт модели в формат, воспринимаемый графическим приложением. Данный этап зависит от выбранного графического движка, наличия дополнительных требований и моделей. Этап может дополнительно включать настройку модели, добавления специфических объектов, таких, как камеры, точки обзора и входа в модель, освещение и т. д.

6. Реализация необходимого функционала в соответствии с требованиями к приложению в среде разработки.

7. Загрузка моделей, полученных на этапе 3, в разработанное приложение, настройка модели, определение точек входа в модель и т. д. На выходе данного этапа получается исполняемый файл и набор ресурсов, представляющий собой конечное приложение.

Наиболее популярными форматами описания трехмерных моделей, которые поддерживаются современными графическими движками, являются OBJ, 3DS, BSP.

Формат файлов OBJ – это простой формат данных, который содержит только 3D-геометрию, а именно позицию каждой вершины, связь координат текстуры с вершиной, нормаль для каждой вершины, а также параметры, которые создают полигоны. Формат OBJ интересен в первую очередь тем, что он прекрасно масштабируется и без искажений в него переносится абсолютно любая модель. При экспорте модели в формат .obj на выходе генерируются 2 файла:

1) file.obj, в котором описаны координаты вершин и ссылка на файл с материалами;

2) file.mtl – material texture library, в котором описываются материалы. Материалом условно называется блок этого файла с описанием координат текстур и ссылками на файлы текстур.

Формат файлов 3DS – это формат, содержащий в себе всю информацию о сцене (данные о геометрии, материалах, текстурных координатах, камерах, анимации и т. д.) в бинарном виде. Данные в файле представляются в XML-формате и не могут быть отредактированы вручную. Формат является основным используемым в продуктах Autodesk 3DS Max и фактически является основным форматом для обмена 3D-изображениями.

BSP является форматом файлов для игр серии Quake и игр, основанных на движке id Tech, таких, как Half-Life и Portal. BSP-файлы используют BSP-деревья (алгоритм двоичного разделения пространства) для упрощения сортировки при рендеринге и уменьшения количества полигонов, одновременно выводимых на экран при каждой перерисовке. В силу ограничений, присущих используемому алгоритму, формат не поддерживает работу с отрытыми пространствами и сценами, не имеющими ограничений. Для создания сцен в формате BSP существуют специализированные редакторы, поддерживающие импорт моделей из форматов OBJ, 3DS.

Из требований к форматам файлов представления графических моделей вытекают требования к средствам разработки трехмерных моделей. Рассматриваемые средства должны иметь либо непосредственную возможность экспортировать модель в форматы OBJ, 3DS, MESH и BSP, либо иметь расширение (плагин), позволяющее это сделать. На сегодняшний день наиболее распространенными программами трехмерного моделирования, имеющими описанные выше возможности, являются Autodesk 3ds Max и Blender.

Autodesk 3ds Max (ранее 3D Studio MAX) – полнофункциональная профессиональная программная система для создания и редактирования трехмерной графики и анимации, разработанная компанией Autodesk. Содержит самые современные средства для художников и специалистов в области мультимедиа. Работает в операционных системах Microsoft Windows и Windows NT (как в 32-битных, так и в 64-битных). Весной 2012 г. выпущена 15-я версия этого продукта под названием «Autodesk 3ds Max 2013».

Blender* – свободный пакет для создания трехмерной компьютерной графики, включающий в себя средства моделирования, анимации, рендеринга, постобработки видео, а также создания интерактивных игр.

Для создания качественной трехмерной модели важно иметь как можно больше информации о создаваемом объекте, поэтому поиск чертежей, схем и фотографий стоит на первом месте. Чертежи необходимы для создания правильной формы объекта и его пропорций, для более подробной детализации модели используются фотографии. Формат исходный данных не имеет принципиального значения, это могут быть как растровые так и векторные форматы, например tga, jpg, png, dwg. После того как материалы найдены, их необходимо импортировать в программу трехмерного моделирования. Затем реализуется процесс моделирования. Важно отметить, что полигонаж модели может быть разным и зависит от поставленных задач, для фотореалистичной визуализации требуется высокополигональная модель, а для менее мощных (например, игровых) движков количество полигонов должно быть минимально. Все модели, используемые в сценах, текстурируются. Это позволяет реалистично имитировать внешний вид поверхностей и отобразить то, что нецелесообраз-

* Официальный сайт продукта Blender. [Электронный ресурс.] URL: http://blender.org.

но делать на этапе моделирования (например, расшивка, надписи и прочие детали, не обладающие объемом либо он незначителен).

Все текстуры можно разделить на две группы:

– текстуры-плитки;

– текстуры-развертки.

Текстура-плитка – многократно повторяющееся изображение, при стыковке не образует видимых швов и служит для покрытия большой однообразной поверхности без ущерба качеству, но с экономией ресурсов системы.

Текстура-развертка рисуется точно под целевую модель с учетом ее формы и пропорций. На такой текстуре изображаются все характерные детали.

Текстуры можно как рисовать с нуля, так и создавать на основе фотографий. Первый вариант требует некоторых художественных навыков. Наиболее популярным редактором для создания и редактирования текстур является Adobe Photoshop.

Любое графическое приложение в среде Linux\Unix работает на нижнем уровне с использованием реализаций функций OpenGL. OpenGL – это спецификация, определяющая независимый от языка программирования платформонезависимый программный интерфейс для написания приложений, использующих двухмерную и трехмерную компьютерную графику. Спецификация включает более 250 функций для рисования сложных трехмерных сцен из простых примитивов. При разработке собственного графического приложения использовать напрямую функции OpenGL нецелесообразно, так как они предназначены только для визуализации и не содержат утилит, позволяющих оптимизировать работу приложения, осуществлять загрузку моделей из файлов, не содержат средств навигации и т. д. При разработке графических приложений используются так называемые графические движки, представляющие собой промежуточное программное обеспечение, главной задачей которого является визуализация трехмерной графики. На сегодняшний день существует множество графических движков, обладающих разным функционалом, производительностью, качеством документации, инструментальных средств. Выбор движка осуществляется на основе требований к графическому приложению, сформулированных на первом этапе разработки. Также от выбранного движка будет зависеть технология разработки трехмерных моделей и экспорта в специфические форматы движка. Этап выбора графического движка является наиболее сложным и трудоемким из вышеперечисленных, так как помимо выбора собственно самого движка он включает портирование движка на целевую операционную систему QNX.

При выборе графического движка следует руководствоваться следующим критериями:

1. Наличие открытого кода. В силу того, что под QNX в открытом доступе не существует собранных графических движков, обычно при выборе движка идет речь о портировании движка с другой платформы. Портирование подразумевает компиляцию исходных кодов под операционную систему (ОС) с учетом специфики платформы: наличия библиотек, примитивов и системных функций ОС.

2. Поддержка работы в ОС семейства UNIX (в идеале работа в QNX). Если в движке не заявлена поддержка Unix-систем, то вероятность портировать его под QNX стремится к нулю.

3. Тип рендеринга, поддержка аппаратного рендеринга (аппаратного ускорения). Существуют движки, поддерживающие только программный рендеринг, а также движки, поддерживающие и аппаратный.

4. Производительность движка, измеряемая в FPS, определяет максимальное количество кадров в секунду.

5. Поддержка функций визуализации только содержимого, попадающего в поле зрение (без визуализации частей моделей за кадром/не видимых на кадре).

6. Возможность работы со сценами: объединение объектов в сцены, динамическая загрузка сцен с целью организации перехода между объектами (к примеру, вход в помещение, переход к более детальному отображению и т. д.).

7. Наличие встроенных функций скрытия/отображения отдельных объектов.

8. Наличие встроенных средств навигации (изменения положения камеры).

9. Наличие контроллеров, отслеживающих нажатия на объекты, выбор манипулятором «мышь» и т. д.

10. Поддержка часто используемых форматов трехмерных моделей типа 3DS, OBJ без дополнительной конвертации.

11. Наличие и качество документации.

Наиболее популярными, с точки зрения указанных критериев, являются следующие графические движки:

id Tech 3;

Irrlicht;

OGRE.

• id Tech 3 – игровой движок, разработанный компанией id Software специально для игры Quake III Arena и с успехом используемый в других проектах. Для данного движка опубликованы исходные коды по лицензии GNU GPL 2.0, последняя версия 1.32 выпущена 19 августа 2005 г. Движок поддерживает ОС Windows, GNU/Linux, Mac OS. Исходный код портированного движка выложен на официальном сайте QNX и доступен для скачивания. Движок написан на языке программирования *C*, не использует дополнительных прослоек, библиотек и т. д. После сборки портированного движка получается непосредственно игра Quake 3 со своими заставками, меню и т. д. Отсутствие комментариев в исходном коде и документации значительно усложняют доработку движка, для которого не известны ни архитектура, ни потенциальные возможности. Отсутствие объектно-ориентированной основы не позволяет прослеживать связи между компонентами движка. Вместе с тем движок может быть без дополнительной доработки адаптирован для задач отображения, в которых не требуется дополнительный функционал. В редакторе GtkRadiant создается сцена, которая загружается в движок. Оператор может перемещаться по сцене, смотреть внешний вид оборудования.

• Irrlicht (Irrlicht Engine*) – трехмерный графический движок, который является бесплатным свободным программным продуктом и распространяется на условиях лицензии zlib. Irrlicht использует возможности OpenGL и нескольких собственных рендереров. Пользователю предоставляются различные функциональные возможности по загрузке и управлению трехмерными (3D) объектами (сцены, модели и т. п.), немногими спецэффектами и графическим интерфейсом пользователя. Не требует подключения сторонних модулей для реализации высокоуровневых функций. Существует 3 официальных дополнения для Irrlicht: IrrKlang (аудиобиблиотека), IrrXML (загрузка и обработка XML-файлов), IrrEdit (редактор сцен). Для использования расширенных функций моделирования физических явлений дополнительно применяется физический движок ChronoEngine. Одна из важных особенностей Irrlicht – его кроссплатформенность, т. е. способность работать на различных платформах. Платформонезависимая прослойка обеспечивает легкую портируемость на различные не поддерживаемые официально платформы, в частности существуют порты под android, iPhone и пр. Последняя версия 1.7.3 (23 февраля 2012 г.). Данный движок основывается на библиотеке SDL (Simple DirectMedia Layer) SDL – это свободная кроссплатформенная мультимедийная библиотека, реализующая единый программный интерфейс к графической подсистеме, звуковым устройствам и средствам ввода для широкого спектра платформ. Данная библиотека активно используется при написании кроссплатформенных мультимедийных программ*. SDL поддерживает большое количество ОС, в том числе QNX. IrrLicht является полностью объектно-ориентированным, имеет большой объем документации, прост в использовании. С аппаратными средствами работает через прослойку SDL, что сказывается на снижении производительности при визуализации трехмерных образов. Также полигоны модели не хранятся в видеопамяти, а каждый раз при визуализации загружаются в нее по шине из оперативной памяти, поэтому движок замедляет воспроизведение сильно нагруженных сцен вне зависимости от параметров видеокарты. Сгладить этот недостаток можно, например, определяя вручную, какие объекты видны данной камере, а остальные объекты делать «невидимыми». Главным недостатком данного движка является требование запуска X-сервера (графической системы, портированной с ОС Linux) для поддержки аппаратного ускорения. При этом запуск X-сервера под QNX при использовании встроенных графических чипсетов Intel невозможен по причине отсутствия реализации совместимого OpenGL agpgart-драйвера. Этот факт свидетельствует о невозможности использования данного движка с аппаратным ускорением.

• OGRE – объектно-ориентированный графический движок с открытым исходным кодом, написанный на *C*++. OGRE изначально создан как графический движок для рендеринга трехмерной графики. Большую популярность движок получил за счет своей гибкости, что позволяет «скрещивать» его со многими другими библиотеками (физика – ODE, Newton, PhysX, Bullet; звук, сеть, графический интерфейс и т. д.). С помощью усилий многих профессионалов появились библиотеки, адаптированные для работы с OGRE. Например, для реализации физики в приложени-

* Официальный сайт продукта Irrlicht. [Электронный ресурс.] URL: http://irrlicht.sourceforge.net/.

* Официальный сайт продукта Simple Directmedia Layer. [Электронный ресурс.] URL: http://www.libsdl.org/.

ях, использующих данный движок, портированы такие библиотеки, как PhysX SDK (NxOgre), Newton Game Dynamics (OgreNewt), Bullet Physics Library (OgreBullet), Open Dynamics Engine (OgreODE). Данный движок зависит от библиотек FreeType2 (http://www.freetype.org/), FreeImage (https://github.com/lynxluna/freeimage), zzlib (http://zziplib.sourceforge.net/), OpenIL (http://openil.sourceforge.net/), которые также должны быть портированы перед портированием движка. OGRE не имеет своей реализации обработки событий клавиатуры и манипулятора «мышь», основываясь на OIS (Object Oriented Input System). Поэтому перехват системных событий требуется реализовать на уровне движка.

В качестве среды разработки приложений для QNX обычно используется IDE Momentics*. В данной среде разработки реализуются этапы кодирования, верификации и сопровождения программного кода. Комплект поставляется в виде трех дистрибутивов, предназначенных для функционирования, соответственно, в среде операционных систем Windows, Linux и QNX Neutrino. Выбор операционной системы для установки QNX Momentics зависит от используемых подходов к разработке и личных предпочтений программистов. QNX Momentics включает в себя:

1. Инструменты для разработки и отладки программ для всех аппаратных платформ, поддерживаемых ОСРВ QNX Neutrino (ARM, MIPS, PowerPC, SH-4, x86).

2. Инструменты для оптимизации программ с помощью прикладного и системного профилирования, анализа памяти и покрытия кода.

3. Инструменты для построения целевых систем QNX Neutrino.

4. Инструменты визуального мониторинга целевых систем QNX Neutrino во время их функционирования.

5. Инструмент для визуального построения графических интерфейсов пользователя – Photon Application Builder.

6. Средства удаленного доступа к рабочему столу Photon microGUI целевой системы QNX Neutrino из сред Windows и UNIX.

Существуют различия при запуске 3D-приложений, разработанных в IDE Momentics в консольном режиме и в графической оболочке Photon. При работе в консольном режиме удаленный запуск приложения может осуществлять только один разработчик, так как консоль имеет один графический контекст, который инициализируется монопольно одним процессом. Таким образом, отладка консольных приложений, использующих 3D-графику, должна производиться разработчиками поочередно или на различных стендах. Запуск и отладка 3D-приложений в графической системе Photon свободны от ограничений консольного запуска, так как графический контекст инициализируется независимо для каждого окна. Поэтому пользоваться одним стендом для отладки могут несколько разработчиков одновременно.

Трехмерные графические приложения ресурсоемки. В связи с этим актуальным является вопрос оптимизации работы графического приложения, который сводится фактически к решению трех задач:

– оптимизация исходной модели;

– оптимизация отображения моделей;

– оптимизация кода приложения.

Оптимизация исходной модели обычно сводится к уменьшению количества полигонов и уменьшению количества объектов. Уменьшение количества полигонов может достигаться снижением детализации малозначимых элементов сцены, вплоть до перевода их на уровень текстур (если объект не значим, то он может быть просто нарисован на текстуре).

Помимо количества полигонов, например для движка OGRE, большое значение имеет количество отображаемых объектов. При подготовке моделей для движка OGRE следует использовать формат mesh. Файл mesh содержит описание одного объекта сцены (корпуса, вентиля, манометра и т. д.). Чем больше отдельных объектов, тем больше файлов с данными, тем медленнее будет рендериться сцена, поэтому рекомендуется при разработке моделей сокращать количество объектов, сокращая таким образом количество mesh-файлов. Сокращение количества объектов может достигаться объединением нескольких объектов в один (к примеру, все переборки корабля, загородки, элемент на палубе могут быть объединены в один объект (mesh), который будет называться «корпус корабля»).

Оптимизация отображения моделей в самом простом случае сводится к снижению детализации отображаемых объектов в зависимости от расстояния до них. Для этих целей обычно используется технология LOD. LOD* (англ. Levels

* Официальный сайт компании «СВД Встраиваемые Системы». [Электронный ресурс.] URL: http://www.kpda.ru.

* Свободная энциклопедия. [Электронный ресурс.] URL: http://ru.wikipedia.org/.

Of Detail – уровни детализации) – прием в программировании трехмерной графики, заключающийся в создании нескольких вариантов одного объекта с различными степенями детализации, которые переключаются в зависимости от расстояния объекта до виртуальной камеры. Смысл приема заключается в том, чтобы не отображать высокодетализированные объекты, находящиеся на большом расстоянии. Другой метод заключается в использовании одной основной, «грубо приближенной», модели и нескольких внешних надстроек к ней. Каждая последующая надстройка к основной модели дополняется элементами детализации пропорционально номеру надстройки, т. е. на самом большом расстоянии будет отображаться единственная главная модель объекта. С приближением же последнего к камере, к конвейеру отрисовки будут последовательно подключаться последующие надстройки деталей. Использование LOD способно существенно снизить требования к ресурсам компьютера при выводе графики на экран. Но в любом случае для использования технологии LOD необходимо разработать несколько вариантов трехмерных моделей. Простейшим вариантом использования технологии LOD, не требующим разработки дополнительных моделей, является простое скрытие удаленных объектов.

Другой способ оптимизации отображения модели заключается в создании единственной текстуры среднего размера с достаточной детализацией и наложении ее на полигоны объекта во всех случаях, когда требуется визуализация. Однако здесь возникают две проблемы. Во-первых, если текстура создана с максимально возможным разрешением и детализацией (например, 2048 × 2048 пикселей при глубине цветового охвата 32 бит), то расход вычислительных ресурсов и памяти при наложении текстур на все видимые объекты в трехмерной сцене будет слишком большим. Во-вторых, если уменьшить размер (и тем самым детализацию) текстур, то объекты по мере приближения к плоскости проекции будут выглядеть все более грубо. Для решения указанных проблем разработана технология, получившая название MIP mapping. Ее суть заключается в предварительном или динамическом создании набора текстур с различным разрешением и уровнем детализации на основе базовой текстуры максимального разрешения. При построении трехмерной сцены определяется удаленность полигона от картинной плоскости и соответственно этому значению накладывается текстура с заданным разрешением. Важно отметить, что, например, графический движок OGRE имеет встроенную возможность автоматически использовать инструменты MIP-mapping.

Современные графические движки позволяют подключать плагины, которые также оптимизируют отображение сложных моделей. Например, графический движок OGRE имеет стандартный плагин под названием Plugin_PCZSceneManager (Portal Connected Zone Scene Manager). Он поставляет особый вид менеджера сцены. Его основная идея заключается в том, что вся сцена разбита на зоны (кубы или шары) и все объекты принадлежат хотя бы одной из зон. Соседние зоны могут быть соединены между собой так называемыми порталами (с геометрической точки зрения – часть поверхности зоны). Порталы всегда существуют парами (если у зоны А есть портал в зону Б, то у зоны Б обязательно есть портал в зону А, имеющий то же расположение). Также зоны могут содержать антипорталы (некие объекты, ограничивающие видимость, которые сами по себе никак не отображаются). Суть в том, чтобы не рендерить невидимые объекты. А видимы они или нет, определяется следующим образом: начиная с зоны, в которой находится камера, движок начинает рендерить все объекты и добавлять в список текущих «перегородок, ограничивающих обзор», антипорталы этой зоны. Далее перебираются все порталы этой зоны и если они не закрыты антипорталами – рекурсивно переходят в зону, куда вел данный портал. Обычно имеется зона по умолчанию (присутствующая всегда), которая содержит все объекты, не привязанные к какой-либо еще зоне. Просто разбить сцену на зоны и соединить их порталами не имеет смысла, пока не добавлены антипорталы, так как, пока их нет, в процессе вывода на экран будут перебираться все зоны. Если сделать зону, в которую не ведет ни один портал, то никакие объекты из нее не будут рендериться (даже если не установлено ни одного антипортала). Так будет, если текущая камера находится вне этой зоны, в противном случае будут видны объекты данной зоны и только они.

Основной сложностью использования данной техники является выбор принципа выделения зон и построения порталов. С одной стороны, объекты могут привязываться к зонам при разработке приложения на языке программирования, что достаточно трудозатратно. Возможен другой под-

ход: в процессе подготовки модели логические объекты, находящиеся за дверьми или перегородками (например, каюты внутри корабля), выделяются прозрачными кубами, имеющими имена с заданным префиксом. Аналогично предлагается поступать с переходами (порталами) – называть двери и перегородки согласованными именами. В таком случае на уровне движка возможно единообразно выделить набор зон и порталов, тем самым обеспечив универсальность при построении моделей и отсутствие задания моделей в коде 3D-приложения.

Оптимизация работы графического приложения может осуществляться тремя способами:

– полностью программная реализация функций OpenGL, при этом все расчеты выполняются центральным процессором с использованием оперативной памяти компьютера;

– частичное выполнение функций графическим микроконтроллером, при этом используется оперативная память компьютера и часть расчетных операций выполняется центральным процессором;

– выполнение всех функций OpenGL на графическом процессоре, при этом в полной мере используется память видеокарты, в нее загружается вся модель и текстуры. Достигается наибольшая производительность, но возрастают требования к объему памяти видеокарты. При этом уменьшаются требования к центральному процессору, так как в случае графических приложений он выполняет обслуживающие функции.

Таким образом, наибольшая производительность работы графических приложений достигается при наличии аппаратного ускорения. Под аппаратным ускорением [см. лит.] понимают применение аппаратного обеспечения для выполнения некоторых функций быстрее по сравнению с выполнением программ процессором общего назначения. При работе с графикой в качестве аппаратного обеспечения применяются графические процессоры (или видеокарты). С аппаратным 3D-ускорением трехмерные изображения создаются графическим процессором видеокарты, при этом не используются ресурсы центрального процессора. Без 3D-ускорения процессор вынужден отрисовывать все самостоятельно, используя библиотеки, которые требуют значительной вычислительной мощности. Основное отличие аппаратного ускорения от программного заключается в параллельности, позволяя аппаратному обеспечению быть гораздо быстрее, чем программное. Аппаратные ускорители специально спроектированы для программного кода, создающего высокую вычислительную нагрузку. В зависимости от степени детализации аппаратное ускорение может варьироваться от небольшой функциональной единицы до крупного функционального блока, как, например, обработка 3D-изображения. Работа с трехмерными графическими изображениями довольно специфична с точки зрения вычислений, поэтому специализированные графические процессоры справляются с этой задачей лучше, чем центральный процессор, предназначенный для решения широкого круга задач. Стандартная поставка QNX предназначена для архитектуры x86 и включает в себя все необходимые драйверы для поддерживаемых аппаратных средств. Для других архитектур встраиваемых систем существуют специальные сборки и так называемые BSP (Board Support Package) пакеты [см. лит.] – интегрированные пакеты драйверов и/или модулей операционной системы, реализующих поддержку определенной аппаратной платформы. По сути BSP является модулем, набором модулей или набором драйверов устройств, встраиваемым в операционную систему и реализующим поддержку всего оборудования и особенностей конкретной аппаратной платформы.

В заключение необходимо отметить, что создание КОС с использованием трехмерных графических отображений дает максимальный эффект обучения и наглядности представления учебного материала, однако требует в процессе разработки внимания ко многим технологическим аспектам. Описанные в статье особенности создания графических приложений в среде QNX необходимо учитывать на всех этапах разработки: формулирование требований к приложению, проектирование пользовательского интерфейса и архитектуры приложения, выбор аппаратной платформы, создание трехмерных моделей, разработка исходного кода приложения, сборка приложения и настройка среды выполнения.

СПИСОК ЛИТЕРАТУРЫ

Кртен Р. Введение в QNX Neutrino. СПб.: БХВ-Петербург, 2011.

E. L. Kalishenko, Yu. V. Romanenko

TECHNOLOGICAL ASPECTS OF REALIZATION OF COMPUTER TRAINING SYSTEMS WITH THREE-DIMENSIONAL GRAPHIC IN QNX ENVIRONMENT

Article covers technological aspects of 3D-applications development as part of computer-based training systems. Graphical applications optimization techniques are considered for QNX realtime operating system.

Computer-based training, simulators, 3D-applications, QNX operating system, graphical applications optimization

УДК 004.41

С. А. Романенко, А. С. Скрипникова

Управление процессом разработки на основании требований

Рассматривается процесс разработки программных изделий, применяемый в открытом акционерном обществе «Научно-инженерный центр Санкт-Петербургского электротехнического университета». Обсуждаются особенности подхода, его достоинства и недостатки, сложности, возникавшие в процессе его внедрения и применения. Подчеркивается положительное влияние выбранного подхода на процесс взаимодействия заинтересованных лиц на всех этапах жизненного цикла разрабатываемых программных изделий.

Домен, матрица трассировки, оптимизация процессов, предметно-ориентированное проектирование, процесс разработки, управление требованиями

С начала апреля 2011 г. в открытом акционерном обществе «Научно-инженерный центр Санкт-Петербургского электротехнического университета» (ОАО «НИЦ СПбЭТУ») ведутся работы по внедрению на уровне предприятия процесса разработки программных изделий на основании требований. Целью внедрения описываемого процесса является оптимизация разработки за счет снижения временных затрат на управление, издержек, связанных с доработкой программных изделий, а также достижения требуемого уровня качества выпускаемых программных изделий.

На сегодняшний день описываемый процесс внедрен в пилотной зоне, в качестве которой выбрана группа проектов по разработке программных продуктов для Федеральной таможенной службы Российской Федерации (ФТС России). В данной статье рассматриваются результаты поэтапного внедрения, приводятся решения возникших проблем.

На момент принятия решения о необходимости внедрения существующий процесс разработки программных изделий не содержал в себе отдельно выделенных подпроцессов сбора, описания требований и управления ими. Поэтому возникла необходимость их выделения и формализации.

В соответствии с международным стандартом IEEE Std 610.12 требованием было принято считать условие или возможность, требуемые пользователем для решения задач или достижения цели; условие или возможность, которые должны удовлетворяться системой (компонентом системы), или которыми система (компонент системы) должна обладать для обеспечения условий контракта, стандартов, спецификаций или других регулирующих документов.

Учитывая требования международного стандарта IEEE Std 830-1998 и особенности существующего на тот момент процесса разработки, в качестве атрибутов требования были выделены:

1. Идентификатор требования (обязательный).
2. Наименование требования (обязательный).
3. Тип требования (обязательный).
4. Описание требования (обязательный).

4.1. Перечень потребителей (обязательный).