України [Електронний ресурс] / Портал: Національна бібліотека ім. В. Вернадського. – Режим доступу \www/ URL: http://www.nbuv.gov.ua/portal/natural/.../15.pdf. – Заголовок з контейнера, доступ вільний, 30.10.2012. **9.** *Володарський, Є. Т.* Метрологічне забезпечення вимірювань і контролю : навчальний посібник / Є. Т. Володарський, *Кухарчук В. В., Поджаренко В. О.* [та ін.]. – Вінниця : ВДТУ, 2001. – 219 с. **10**. *Харченко, О. Г.* Методи забезпечення та контролю якості Web-застосувань на стадіях життєвого циклу [Електронний ресурс] / *О. Г. Харченко, В. В. Яцишин, І. О. Боднарчук* // Портал: Національна бібліотека ім. В. Вернадського. – Режим доступу \www/ URL: http://www.nbuv.gov.ua/portal/...1/34har.pdf. – Заголовок з контейнера, доступ вільний, 30.10.2012. **11.** *Новицкий, П. В*. Динамика погрешностей / *П.В. Новицкий, И.А. Зограф, В.С. Лабунец*. – Л.: Энергоатомиздат. Ленингр. отд-ние, 1990. – 192 с.: ил. **12.** *Мельницкая, Ж.С.* Теоретическое и экспериментальное исследование процесса старения электроизмерительных приборов / Автореферат дис. на соиск. учен. степени канд. техн. наук, 1970. – 16 с. (Ленинградский политехнический институт).

Надійшла до редколегії 20.03.2013

УДК 681.2.088 : 681.518:629

Аналіз показників якості інформаційно-вимірювальних систем/ Грабовський О. В. // Вісник НТУ «ХПІ». Серія: Нові рішення в сучасних технологіях. – Х: НТУ «ХПІ», – 2013. - № 1 (977). – С. 59-66. – Бібліогр.:12 назв.

Рассматриваются показатели качества в общей модели жизненного цикла информационно-измерительной системы в целом и ее отдельных компонентов. Показано, что показатели должны определяться в техническом задании согласно действующих норм. Сделан акцент на том, что показатели качества на каждом этапе жизненного цикла являются комплексными показателями, которые состоят из соответствующих показателей технических и программных средств и решений.

Ключевые слова: информационно-измерительная система, качество, показатель, мониторинг, жизненный цикл.

Considered quality in the general model of the life cycle of information-measuring system. Gives figures for the whole system and the performance of individual components. It is shown that the indicators should be defined in the terms of reference under the current regulations. Emphasis on the fact that the quality at every stage of the life cycle are complex indicators. It is shown that they are composed of indicators of hardware and software tools and solutions.

Keywords: information-measuring system, quality indicators, monitoring, life cycle.

УДК 004.358:681.518

М. А. КАРНАУХОВ, бакалавр, ХНУРЭ, Харьков

ПРИНЦИПЫ РАЗРАБОТКИ ГРАФИЧЕСКО-ЛОГИЧЕСКОГО ДВИЖКА ДЛЯ ИГРОВЫХ ПРИЛОЖЕНИЙ НА ОС ANDROID

Разработка мобильных приложений, конечно, имеет много общего с разработкой программного обеспечения для настольных компьютеров. И развитие мобильных приложений включает следующие шаги: анализ предметной области, выявление функциональных требований к приложению, разработка движка, развитие пользовательского интерфейса, разработка моделей данных, развертывание мобильных приложений.

Ключевые слова: программное обеспечение; модель; движок; анализ.

Введение. В контексте широкого использования мобильных устройств и придания им статуса предметов первой необходимости для активной социальной жизни и досуга современного поколения, одним из самых актуальных направлений в разработке программного обеспечения, стало являться написание программ для планшетных ПК, смартфонов и т.д., в частности игровых приложений.

Современные представители мира мобильных устройств, такие как планшеты, коммуникаторы и смартфоны – это устройства с расширенной функциональностью, по сравнению с обычным мобильным телефоном, включающей в себя широкий спектр мультимедийных программных функций, и имеющий разветвлённую операционную систему(в отличии от прошивки присутствующей в обычных мобильных телефонах),

открытую для разработки программного обеспечения сторонними разработчиками. В связи с потребностью пользователей возникшей к многозадачным устройствам - рынок средств мобильной связи переживает настоящий бум «умных» устройств. Их качество в большей степени зависит от аппаратных характеристик, а вот удобство пользования в большей степени зависит от мобильной операционной системы.

Обращая внимание на вышеизложенное, следует учитывать, что миллионы вышеупомянутых устройств работает посредством операционной системы Android. В связи с высоким спросом на программные разработки под данную платформу, исследования методов и подходов к написанию программ для указанной операционной системы, а также создание насыщенных игровых приложений ориентированных на пользователей ОС Android, с использованием технологий, сопряженных с этой платформой, являются на сегодняшний день актуальными и требуют особого внимания.

Анализ предметной области. Человеческие потребности быть всегда в курсе дел выводят информационные технологии на прямую линию по созданию все новых девайсов и гаджетов. Неудобство эксплуатации компьютеров и ноутбуков обуславливает появление различных мини-компьютеров, смартфонов и коммуникаторов, в основе которых лежит все та же операционная система. Лидирующие позиции на сегодняшний день занимают платформы Android и iPhone, но им ничем не уступает по популярности и Windows, ранее Mobile, а сегодня уже Windows Phone 7. Но все эти платформы могут работать полноценно только при одном условии - если была для них осуществлена разработка мобильных приложений. Все мобильные приложения делятся на несколько категорий, но в данном контексте нас интересуют приложения относящиеся к развлекательным программным продуктам, а именно игровые приложения. Учитывая указанные данные, особое внимание стоит уделить графической составляющей игр, то есть игровым движкам. Игровой движок - это центральный программный компонент компьютерных и видео игр или других интерактивных приложений с графикой, обрабатываемой в реальном времени. Он обеспечивает основные технологии, упрощает разработку и часто даёт игре возможность запускаться на нескольких платформах, таких как игровые консоли и настольные операционные системы, например, Linux, Mac OS X и Microsoft Windows. Основную функциональность обычно обеспечивает игровой движок, включающий движок рендеринга («визуализатор») для 2D или 3D графики, физический движок или обнаружение столкновений (и реакцию на столкновение), звук, скриптинг(система скриптования), анимацию, искусственный интеллект, сетевую поддержку, работу с потоками, управление памятью и граф сцены.

Цель и задачи исследования. Целью работы является исследование принципов организации архитектуры графическо-логической составляющей движка игровых приложений для мобильных устройств под операционную систему Android.

Разработка метода представления нечетких систем средствами реляционных баз данных. В виду необходимости разработки независимой технологической основы для быстрой разработки на базе нее пользовательских игровых приложений трехмерной визуализации в реальном времени. Иными словами, как было определено выше «движок». Прежде всего стоит определить приоритеты в контексте данных разработок. Конечный продукт должен обладать следующими качествами: легкая расширяемость без необходимости пересборки всех исходных кодов; гибкость, возможность использования в различных разноплановых проектах, поддержка современных графических технологий, кроссплатформерность; несколько уровней абстракции; четкая ООП модель; поддержка нескольких языков программирования, стабильность в работе.

Для того чтобы спроектировать по-настоящему гибкую и масштабируемую систему игры, необходимо не только понимать специфику игровых приложений, но и иметь представление об архитектуре программного обеспечения в общем.

Удобная архитектура должна не только логически разделять систему на подсистемы, она также должна предусматривать легкую замену или модификацию любой подсистемы без перестраивания системы целиком.

Отчасти проблемы заключаются в том, что в основе практически каждой модели игры лежит принцип объектно-ориентированности и, следовательно, в каждой игре присутствует чрезмерное количество взаимосвязей между компонентами системы. Игры практически всегда имеют дело с объектами, живущими в виртуальном мире. Игровые объекты обладают собственным поведением, отрисовывают сами себя на экране, а иногда даже говорят сами. Такой подход кажется логичным и его распространение, по всей видимости, связано со всеобщим признанием объектно-ориентированной парадигмы. Однако ограниченность данного подхода проявляется, как только сложность и запутанность таких функций как отрисовка (rendering) и принятие решений (ИИ) начинает расти экспоненциально. В свою очередь данное усложнение приводит к громоздкости игровых объектов в частности и архитектуры вообще.

Переход к системам, построенным на COTS (COTS - component off the shelf, методология, предписывающая разработчикам использование сторонних компонентов(плагинов)), является первым шагом на пути масштабного изменения игр, принципов их разработки и планирования. Однако несмотря на то, что использование готовых компонентов способно улучшить качество игры и сократить время ее разработки, использование объектно-ориентированного подхода приведет к тому, что компоненты едва ли будут более полезны, чем обычные библиотеки функций, помогающие оперировать объектами. Игровые объекты все еще ответственны за обработку своих данных, а именно, за рендеринг, за искусственный интеллект, звук, физику и т.д. Одним словом, игры пытаются использовать преимущества COTS-компонентов и вместе с этим оперируют слишком большими и сложными игровыми объектами. А это в свою очередь приводит к тому, что при разработке объектов разработчики жестко привязаны к используемым COTS-компонентам (рис. 1).

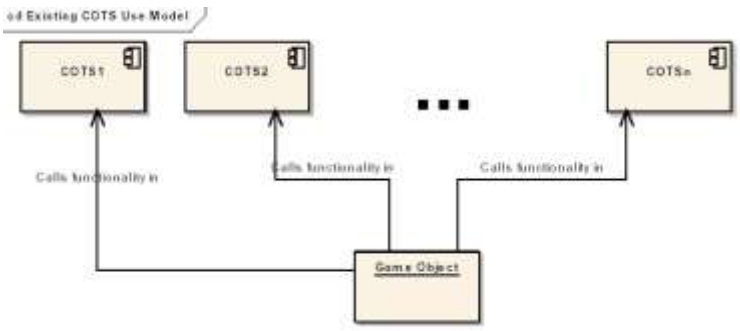

Рис 1 - Сочетание метода COTS-разработки и объектно-центрированной модели

Вдобавок объектно-ориентированный подход ограничивает возможность повторного использования, даже при применении технологии COTS-компонентов. Код, описывающий объект, всегда остается наихудшим в смысле повторного использования, но при совмещении объектно-ориентированных и COTS-технологий этот код содержит в себе еще и вызовы функций различных специализированных компонентов. Таким образом, при переходе на новый проект разработчики часто вынуждены переписывать код вследствие подобного взаимодействия объекта и компонентов. Даже хорошо спланированная и разработанная иерархия классов не способна уменьшить риск переписывания кода из-за того, что объекты остаются жестко связанными с используемыми COTS-компонентами. Я полагаю, что существует архитектура, способная значительно сократить количество переписываемого кода, вследствие ликвидации жестких взаимосвязей.

Спроектировать архитектуру программного обеспечения как достаточно расширяемую, чтобы соответствовать современным технологиям, так и достаточно гибкую, чтобы поддерживать различные типы игр и приложений. Подобная архитектура обеспечила бы надежный фундамент для дальнейшего развития и, возможно, стала бы началом стандартизации взаимодействия между компонентами, используемыми в игре.

Удачная архитектура развивалась и расширялась бы вместе с увеличением сложности современных игр и при этом не ограничивала бы творческий подход разработчиков игр. Для того чтобы достичь этой высокой цели, итоговая архитектура должна отвечать определенным требованиям.

Поддержка концепции COTS-компонентной разработки. Во-первых, архитектура должна иметь строгую логическую структуру. Необходимо таким образом разделить систему на логически независимые части, чтобы каждая из подсистем могла быть разработана и оттестирована независимо от других. Это требование согласовывается с требованиями к COTS-системам, и я буду придерживаться принципов COTS-концепции.

Для того чтобы проверить, отвечает ли полученная архитектура данному требованию, разработчик должен продемонстрировать, что компоненты системы были разработаны и оттестированы независимо.

Сокрытие деталей реализации и доступность применения без знания предметной области (локализация предметных областей). Архитектурное требование о сокрытии деталей реализации и доступности применения существует из-за различных возможностей, требуемых в различных играх. В современных играх требуется высококлассная графика, реалистичная физика, человекоподобный искусственный интеллект, звук кинематографического качества. Даже если игровой разработчик использует COTS-компоненты для создания всего этого в игре, он должен хорошо знать предметную область - для того, чтобы использовать конкретный компонент корректно.

Гибкость и модифицируемость. Гибкость - ключ к будущему игр. Из-за постоянно растущей стоимости разработки возможность комбинировать и изменять существующие компоненты, критична в плане сдерживания цен. Предлагаемая архитектура не зависит ни от жанра игры, ни от технологий, воплощаемых в ней, что позволяет разработчикам создавать различные игры, используя различные технологии. Для того чтобы проверить, отвечает ли полученная архитектура данному требованию, достаточно продемонстрировать, что разные игры могут быть написаны на ее основе.

Масштабируемость и эксплутационная надежность. Другими критическими требованиями к архитектуре, также вытекающими из проблемы быстрого роста стоимости разработки, являются масштабируемость и эксплутационная надежность. Успешные игры часто появляются заново с различными дополнениями и улучшенным качеством. Для того чтобы проверить, отвечает ли полученная архитектура данному требованию, достаточно продемонстрировать, что архитектура с легкостью поддерживает новые или улучшенные технологии, а также позволяет легко добавлять новые функциональные схемы в игру. Производительность и другие качественные параметры не являются приоритетными.

Может показаться странным, что я не считаю производительность ключевым, главным требованием при проектировании архитектуры программ, предметная область которых постоянно напоминает о высокой планке производительности. Причина в том, что производительность в действительности играет несравнимо меньшую роль в вопросах межкомпонентного взаимодействия, чем производительность самих компонентов и подсистем игры. Например, отрисовка 10 миллионов треугольников одного объекта гораздо сильнее влияет на производительность, чем единичное межкомпонентное воздействие на графическую систему, вызывающее эту отрисовку.

В движке есть четко разграниченные уровни абстракции, которые по принципу слоеного пирога логически разделяют программирование для разных уровней движка. Выше уже писалось, что такое разделение необходимо, во-первых, что бы была четкая логическая структура движка в целом, во-вторых, что бы пользователь не мог получить прямой доступ к любому места движка (фактически доступ он конечно имеет, но что бы пойти и переправить что - либо в рендере например это уже должен быть обдуманный шаг и скорее всего пользователь постарается изначально решить эту проблему на более

высоком уровне абстракции), в-третьих, это разграничение полезно самим разработчикам движка.

Идеология движка заключается в том, что пользователь не получает исходные коды движка, в его распоряжении оказывается лишь набор интерфейсов, сам движок - набор библиотек и документация к интерфейсам. По этому изначально пользователю доступно лишь несколько верхних уровней абстракции, о которых я напишу чуть ниже.

Программирование на каждом из уровней абстракции осуществляется на одном или нескольких пригодных для этого уровня языках программирования. На самом верхнем уровне это скриптовый интерпретируемый движком язык, а на нижнем это может быть только C++. Схема приведена на рисунке 4 на следующей странице.

Пользователь получает доступ к следующим уровням.

Самый верхний, как это уже было написано выше это скриптовый язык. Он представляет собой небольшой набор классов с основным функционалом и не позволяет с этого уровня этот функционал изменять, тем не менее, на этом уровне без проблем можно польностью реализовать несложное приложение. Основным плюсом работы на этом уровне является то, что пользователь работает с интерпретируемым языком со всеми его плюсами, а следовательно разработка кода наиболее проста и эффективна.

Выводы. На основе анализа особенностей организации архитектуры графическо-логической составляющей движка игровых приложений для мобильных устройств под операционную систему Android были определены основные концепции, а также выделено приоритетное направление для разработок в данном направлении.

Список литературы: 1. *Заде, Л. А.* Основы нового подхода к анализу сложных систем и процессов принятия решений [Текст] /*Л. А. Заде*. // Математика сегодня. - М. : Знание, 1974,- С. 5–49. **2.** *Казман Р. 3* миллиона строк кода [Текст]/ *Р. Казман* / М.: Знание, 2004,- с. 173**. 3**. *Клементс П.* Архитектура программного обеспечения на практике [Текст] / П. Клементс / М.: Мир, 1976,- с. 241. **4**. *Гаврилова, Т. А*. Базы знаний интеллектуальных систем [Текст] / *Т. А. Гаврилова, В. Ф. Хорошевский* – СПб. : Питер, 2000. – 301-350 с. 5. Базы данных. Интеллектуальная обработка информации/ [Текст] / *В. В. Корнеев, А. Ф. Гарев, С. В. Васютин, В. В. Райх* . – М. : Знание, 2000. – 251 с. **6**. *Конноли Т., Бегг К.* Эссе в объектно-ориентированной разработки программного обеспечения/ [Текст] /Т. Конноли, К. Бегг. – М. : Знание, 1992. – 432 с**. 7**. *Фристром Д.* (2004). Менеджер в чужой стране: большинство проектов провалено[Текст] /*Т. Конноли, К. Бегг*. – М. : Знание, 2004. – 511 с. 8. *Моррис Д.* Архитектура и дизайн игр. / [Текст] / *Д. Моррис*/ М.: Мир, 2000,- с. 367. **8.** *Родженд В*. Базы данных и 3Д движки: [Текст] / *В. Родженд* – СПб. : Питер, 2000. – 384 с.

Надійшла до редколегії 20.03.2013

УДК 004.358:681.518

Принципы разработки графическо-логического движка для игровых приложений на ОС ANDROID/ Карнаухов М. А. // Вісник НТУ «ХПІ». Серія: Нові рішення в сучасних технологіях. – Х: НТУ «ХПІ», – 2013. - № 1 (977). – С. 66-70. – Бібліогр.: 8 назв.

Розробка мобільних додатків, звичайно, має багато спільного з розробкою програмного забезпечення для персональних комп'ютерів. І розвиток мобільних додатків включає наступні кроки: аналіз предметної області, виявлення функціональних вимог до додатка, розробка движка, розвиток користувальницького інтерфейсу, розробка моделей даних, розгортання мобільних додатків.

Ключові слова: програмне забезпечення; модель; движок; аналіз.

The given work is devoted to the modern developments in the field of mobile game programs. Development of mobile applications, of course, has a lot to do with the development of software for desktop computers. And the development of mobile applications has the following steps: domain analysis, the identification of the functional requirements of the application, engine development, user interface development, development of data models, deployment of mobile applications.

Keywords: Software; Model; Engine; Analysis.