М. Календер, П. Б. Панфилов, Сравнительный анализ подходов к разработке приложений интерактивной 3-мерной визуализации,
ИТиВС, 2009, выпуск 3, 27–32

Сравнительный анализ подходов к разработке приложений интерактивной 3-мерной визуализации

М. Календер, П.Б. Панфилов

Аннотация. Представлены результаты экспериментальных сравнительных исследований производительности пяти популярных библиотек функций 3-мерной компьютерной графики: OpenGL, JOGL, OpenSceneGraph, JavaOSG и Java3D. Для проверки производительности графики были разработаны тестовые программы, которые выполнялись в разнообразных конфигурациях с различной сложностью и режимами рендеринга виртуальной сцены. По результатам экспериментов предлагаются рекомендации разработчикам приложений интерактивной 3-мерной визуализации.

Ключевые слова. Интерактивная 3-мерная визуализация, виртуальные среды, графические библиотеки, графы сцены.

Введение

Графические библиотеки (библиотеки функций 3-мерной компьютерной графики) могут быть условно сгруппированы в два больших класса: низкоуровневые иерархические программные интерфейсы к графическому оборудованию (ускорителям 3-мерной графики) и высокоуровневые объектно-ориентированные структуры данных, задающие графы сцены. Примерами наиболее популярной «свободно доступной» низкоуровневой графической библиотеки является библиотека OpenGL [1], а высокоуровневых графов сцены – библиотеки OpenSceneGraph [3] и Java3D [4].

Интерфейс прикладного программирования (API) OpenGL обычно описывается в терминах языка программирования C/C++, хотя существуют «привязки» библиотеки к ряду других языков, таких как Ada, Java, Pascal и Delphi [1, 2]. Библиотека функций графа сцены OpenSceneGraph представляет собой API-интерфейс для языка C++, построенный на базе низкоуровневой библиотеки OpenGL, и имеет также «привязку» к языку Java [3]. API-интерфейс Java3D – это сложная полно-функциональная система рендеринга 3-мерной графической и звуковой среды на языке программирования Java [4, 6].

Правильный выбор между этими библиотеками чрезвычайно важен для успеха в реализации проекта системы интерактивной 3-мерной визуализации, так как у каждой библиотеки есть свои преимущества и недостатки и каждая из них может оказаться лучшей (по параметрам времени и стоимости разработки, производительности и качества приложения) в зависимости от конкретной ситуации конкретного приложения. Чтобы лучше понять, в каких ситуациях какую библиотеку лучше использовать, было выполнено сравнительное исследование графических библиотек на основе тестового приложения визуальной имитации. В качестве объектов исследования были выбраны библиотеки OpenGl, JOGL (которая представляет собой "привязку" библиотеки OpenGL к языку Java), OpenSceneGraph, JavaOSG («привязка» библиотеки OpenSceneGraph к языку Java) и Java3D.

Эксперимент

Специально для целей нашего исследования был разработан инструментальный экспериментальный комплекс, позволяющий собирать разнообразные данные по результатам прогона тестового приложения, необходимые для анализа и сравнения различных реализаций приложения по множеству параметров (структурных, емкостных, временных, производительности и др.).

Экспериментальный стенд. В качестве базовой вычислительной платформы для осуществления экспериментальных прогонов тестового приложения 3-мерной визуальной имитации использовалась IBM PC-совместимая персональная ЭВМ со следующими характеристиками: 1.8-Гигагерцовый процессор AMD Athlon 64 Processor 3000, 2048 Мбайт оперативной памяти, ускоритель 3-мерной графики NVIDIA GeForce 6600 с 256 Мбайтами графической памяти, операционная система MS Windows XP. Тестовое приложение представляет собой 3-мерную интерактивную визуальную имитацию, которая запускается в различных конфигурациях для сбора значений таких параметров, как частота кадров (частота обновления графики) и потребление памяти. Тестовое приложение было реализовано в пяти версиях с использованием следующих графических библиотек: OpenGL version 2.0.3, JOGL version 1.1.0, OpenSceneGraph version 1.2, JavaOSG version 0.33 и Java3D version 1.5.0 beta2.

Тестовое приложение «Кубический вихрь». Динамическая виртуальная среда, получившая условное название «Кубический вихрь», используемая в сравнительном исследовании для сбора данных о производительности визуальных приложений, представляет собой множество кубических объектов, сделанных из различных материалов, разбросанных случайным образом в ограниченном 3-мерном пространстве и вращающихся вокруг общей оси (вертикальной оси Y) (Рис. 1).

Сложность виртуальной среды, определяемая количеством объектов-кубиков (или полигонов), может варьироваться в широких пределах. Для сбора «разумно достаточного объема» статистики по параметрам производительности тестового приложения было принято решение

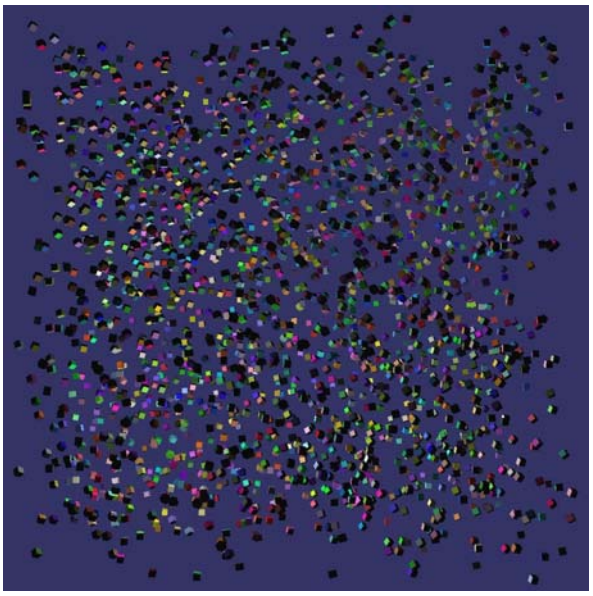


Рис. 1. Тестовое визуальное приложение «Кубический Вихрь»

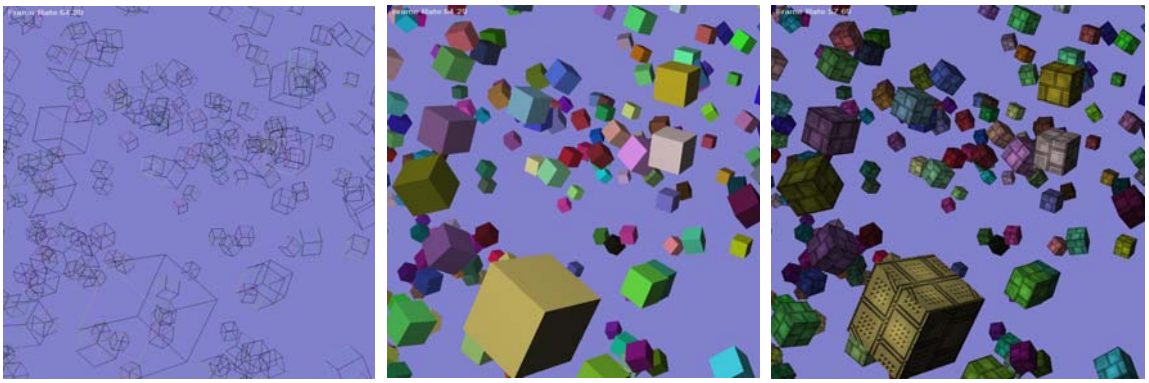
ограничиться 8-ю типоразмерами сложности виртуального пространства, как это представлено в таблице.

Параметры тестового приложения	
Число кубиков в среде	150, 300, 600, 1200, 2400, 4800, 9600, 19200
Режим рендеринга	«Проволочный», Равномерная закраска, Текстурированный
Реализация (Графическая библиотека)	OpenGL, JOGL, OpenSceneGraph, JavaOSG, Java3D

Виртуальная среда «Кубический вихрь» не является интерактивной, но позволяет пользователю задавать различные режимы рендеринга, такие как «проволочная модель» (Wireframe), «равномерная закраска» (Flat shaded или SunLighted) и «текстурированная модель» (Textured) (Рис. 2 (а, б, в)).

Результаты экспериментов

Сравниваемые графические библиотеки тестировались с использованием пяти реализаций тестового приложения в 120-ти различных конфигурациях. Каждая конфигурация прило-



а) «проволочный» б) равномерной закраски в) текстурированный

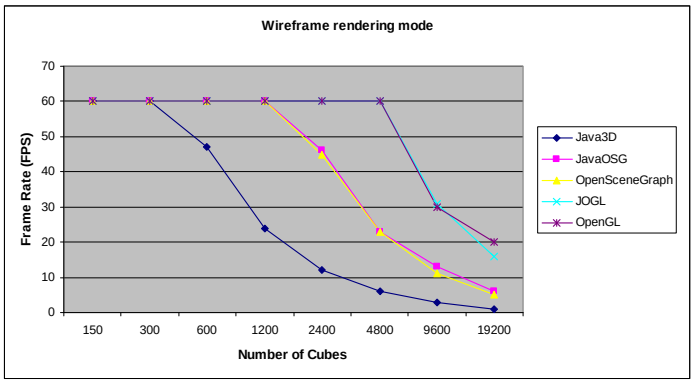
Рис. 2. Режимы рендеринга виртуальной среды

жения прогонялась в течение одной минуты на базовой вычислительной платформе. В специальных файлах результатов испытаний оперативно накапливались данные частоты обновления графики (число кадров в секунду) и потребляемой оперативной памяти.

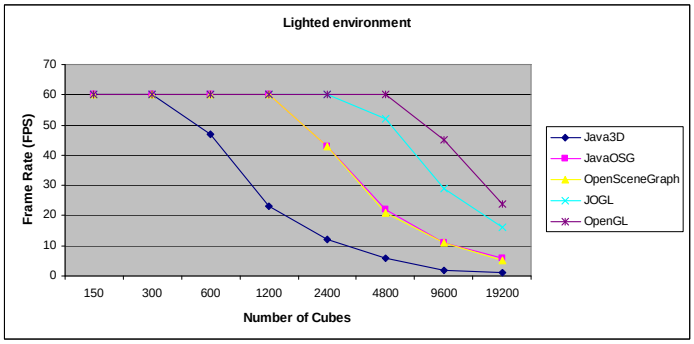
Частота обновления графики. Рис. 3 (а, б, в) показывают кривые зависимости параметра средней частоты обновления графики для конфигураций, отличающихся только по сложности сцены (числу кубиков). Результаты рендеринга сцен в различных режимах (проволочном, равномерной закраски и текстурированном) оказались аналогичными, и во всех случаях наилучшие результаты по производительности приложения в зависимости от сложности сцены оказались у реализации тестового приложения на языке C++ с библиотекой OpenGL, тогда как реализация на базе библиотеки Java3D оказалась наихудшей по этому показателю. Как и ожидалось, низкоуровневый графический API-интерфейс OpenGL показал лучшие результаты, чем высокоуровневые объектно-ориентированные графы сцены. Причем версия приложения на базе библиотеки OpenGL на языке C++ показала лучшие по сравнению в версией на языке Java результаты в экспериментах, так как язык Java оказался «медленнее», чем C++.

Интересным результатом экспериментов можно признать то, что реализация на базе JavaOSG показала примерно такие же результаты, как и реализация на базе OpenSceneGraph, хотя язык C++ «быстрее» языка Java. И наконец, реализация на базе Java3D показала наихудшие результаты среди всех реализаций приложения как на базе графических бибилиотек, так и на базе графов сцены. Данные результаты экспериментальных исследований можно рассматривать как хорошее объяснение того факта, что достаточно популярный (особенно в университетских лабораториях) проект графа сцены «с открытым кодом» Java3D в последнее время теряет поддержку со стороны пользователей и разработчиков, а фирма Sun поддерживает проект библиотеки JOGL.

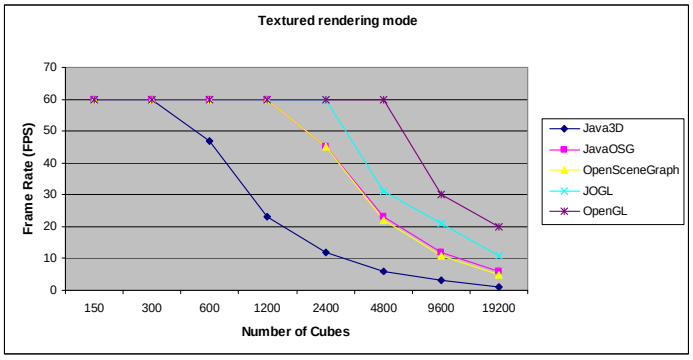
Зрение человека весьма чувствительно к проявлениям дискретности динамической виртуальной сцены как при рендеринге отдельных объектов (таких как «граненость» полигонной апроксимации поверхностей, «бегущий» алиасинг ребер полигонов), так и при воспроизведении анимаций и динамики изменений виртуальной среды («скачкообразное» изменение вида сцены). Для обеспечения максимального реализма динамической виртуальной среды разработчики визуальных имитаций стремятся обеспечить стабильную частоту обновления графики на уровне не меньше 30 кадров в секунду или даже 60 кадров в секунду (если предполагается использование стереоскопических дисплейных систем). Такая частота обновления обеспечивает визуальную «плавность» и непрерывность протекания динамических процессов в виртуальной среде. Это особенно важно для таких серьезных приложений, как имитация внешней визуальной обстановки в тренажерах авиационного и космического назначения, наземных и других транспортных средств, медицинских системах и прочих.



а)



б)



с)

Рис. 3. Средняя частота обновления графики как функция сложности виртуальной среды при разном режиме рендеринга

а) «проволочном»
б) равномерной закраски
в) текстурированном

Приведенные результаты экспериментов с тестовым приложением можно использовать (с рядом оговорок) для поиска ответа на вопрос, какая графическая библиотека лучше всего подходит для реализации визуального приложения заданной сложности с требуемым показателем производительности для поддержания удовлетворительного уровня реализма виртуальной сцены. Таким образом, они могут быть положены в основу выработки рекомендаций разработчикам визуальных имитаций по выбору графической библиотеки для конкретного проекта.

Однако в этой связи стоит отметить, что хотя высокоуровневые библиотеки графов сцены, такие как Java3D, OpenSceneGraph и JavaOSG, и проигрывают низкоуровневым графическим API-интерфейсам в соревновании на «чистую» производительность визуальной имитации, они способны обеспечить (в качестве составной части ядра механизма графа сцены) такие дополнительные средства улучшения общей производительности интерактивной 3-мерной визуальной имитации, как «усечение элементов сцены, находящихся вне пирамиды видимости», «усечение элементов, перекрываемых другими элементами в поле зрения», «усечение элементов, несущественных для визуальной сцены», а также механизмы «уровня детализации», сортировки состояний виртуальной имитации, массивов вершин и дисплейных списков. Таким образом, если нет необходимости показывать одновременно все объекты виртуальной сцены, то появляется возможность построения с помощью механизма графа сцены более сложных виртуальных сред.

Потребление памяти. Рис. 4 показывает графики зависимости объема потребляемой оперативной памяти тестового приложения от сложности виртуальной среды «кубического вихря» (т.е. числа кубиков) для различных конфигураций приложения. Лучшие показатели потребляемой памяти в функции от сложности сцены оказались снова у реализации приложения на базе библиотеки OpenGL на языке C++, тогда как худшие результаты показала реализация на базе графа сцены Java3D.

Если принять во внимание особенности используемого языка программирования и реализации графических библиотек, то следует заметить, что язык Java использует больше оперативной памяти, чем язык C++, и для реализации механизмов графа сцены требуется

больше памяти, чем для библиотечных функций OpenGL, прежде всего, из-за объектно-ориентированной структуры механизма графа сцены. Согласно результатам проведенных экспериментов реализация тестового приложения на базе Java3D столкнулась с проблемами нехватки оперативной памяти для приложения, когда сложность виртуальной сцены превысила величину 4800 кубиков. Хотя на самом деле, возможно, это и не так, просто при сложности сцены выше указанной величины реализация на базе Java3D показала недостаточную производительность (частоту обновления графики), т.е., потребление памяти в данном конкретном случае, возможно, и не является определяющим параметром.

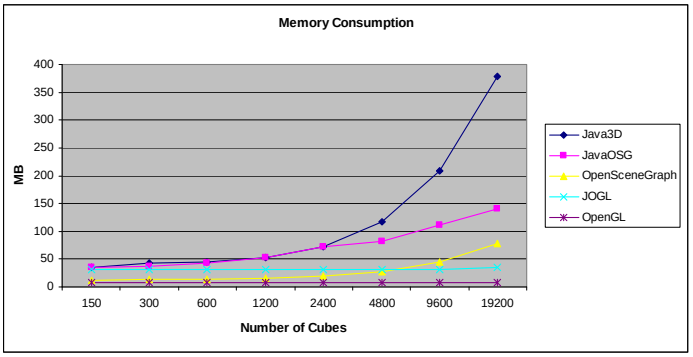


Рис. 4. Потребляемая память в зависимости от сложности сцены

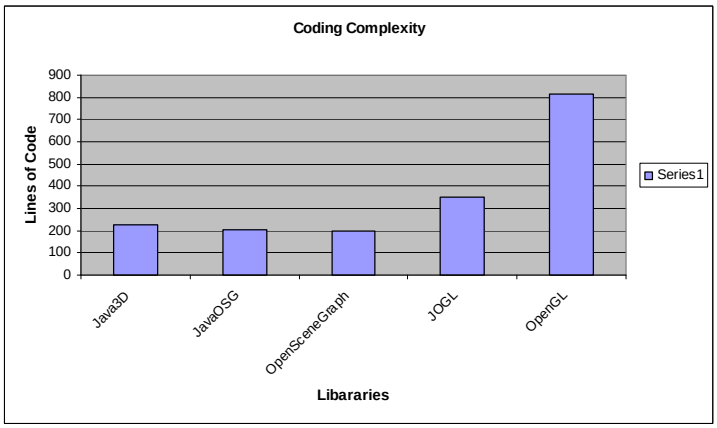


Рис. 5. Сложность программного кода в зависимости от используемой графической библиотеки

Другие аспекты. Помимо таких характеристик, как производительность и потребляемая память, при разработке сложного визуального приложения, есть и другие важные для выбора графической библиотеки аспекты, например, время и стоимость разработки.

Низкоуровневая графическая библиотека OpenGL содержит только базовые функции компьютерной графики, что требует от разработчика самостоятельно проектировать и программировать такие функции, как определение коллизий объектов, файловые загрузчики и взаимодействия пользователя с виртуальной средой. В то же время, библиотеки графов сцены уже содержат подобные высокоуровневые функции, что значительно облегчает программирование приложений, сокращает время разработки и уменьшает вероятность внесения ошибок в проект. Таким образом, если временные ресурсы реализации проекта ограниченны и сложность моделируемого пространства управляема, то имеет смысл рассматривать библиотеки графов сцены в качестве основы реализации проекта.

Еще один важный аспект для сравнения реализаций визуальных приложений составляют особенности языков программирования, таких как, например, Java и C++. Язык Java обладает определенными преимуществами своего высокоуровневого API-интерфейса - это средства файловых загрузчиков, поддержки сетевых соединений и звука. Более того, программы на языке Java могут выполняться в рамках веб-браузера, что позволяет легко и просто создавать Интернет-приложения. С другой стороны, язык C++ предлагает разработчику преимущества высокой производительности приложения и поддержки (графического) оборудования на низком уровне.

И наконец, на Рис. 5 показана зависимость сложности программного кода приложения от используемой графической библиотеки. В силу первой из причин, отмеченных выше, реализации приложения на базе библиотек JOGL и OpenGL демонстрируют более длинные коды, чем реализации на базе графов сцены. Вторая причина объясняет более длинный код реализации на базе OpenGL по сравнению с JOGL. Примерно 500 строк кода реализации тестового визуального приложения на базе библиотеки

OpenGL отвечают только за операцию чтения текстового файла. В случае же использования других библиотек те же самые функции реализуются с помощью двух-трех строк кода.

С точки зрения используемой среды программирования, язык Java потребляет больше памяти компьютера, чем язык C++, и, в свою очередь, библиотеки графов сцены требуют больше памяти, чем библиотека OpenGL, главным образом, из-за реализованных в них объектно-ориентированных структур графа сцены. Как отмечалось выше, результаты тестовых прогонов реализации на базе графа сцены Java3D показали проблемы с памятью при превышении сложности виртуальной сцены 4800 кубиков. Это, скорее всего, является прямым следствием некорректной реализации механизмов графа сцены в проекте библиотеки Java3D.

Заключение

Сравнение производительности графических библиотек, как это представлено в предыдущем разделе, кажется, свидетельствует о том, что наихудшим выбором для реализации приложения 3-мерной визуальной имитации является библиотека Java3D. Хотя само по себе утверждение, что какая-то библиотека является лучше другой во всех случаях, является достаточно спорным. При выборе графической библиотеки необходимо четко представлять себе требования прикладной разработки и осуществлять выбор библиотеки, наилучшим образом удовлетворяющей этим требованиям. Для приложений с высокой сложностью сцены можно рекомендовать использовать библиотеки OpenGL и JOGL. Кроме того, в случае необходимости использования графических функций низкого уровня правильным выбором является библиотека OpenGL. И наконец, в случае разработок с ограниченным бюджетом как по времени, так и по стоимости, оптимальный выбор представляют библиотеки OpenSceneGraph и JavaOSG.

Литература

1. OpenGL – The Industry's Foundation for High-Performance Graphics, http://www.opengl.org/.
2. The JOGL API Project, https://jogl.dev.java.net/.
3. The OpenSceneGraph Project, http://www.openscenegraph.com/.
4. Sun Microsystems Inc. Sun Developer Network's Java 3D API Project, http://java.sun.com/products/java-media/3D/ .
5. A NeHe OpenGL tutorials, http://nehe.gamedev.net/
6. Sun Microsystems Inc. Java 3DTM Documentation, 1999-2000.
7. Java 3D organization tutorials, http://www.java3d.org/tutorial.htm.
8. K. Elissa, "An Overview of Decision Theory," unpublished. (Unplublished manuscript)
9. R. Nicole, "The Last Word on Decision Theory," J. Computer Vision, submitted for publication. (Pending publication).
10. C. J. Kaufman, Rocky Mountain Research Laboratories, Boulder, Colo., personal communication, 1992. (Personal communication)
11. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron Spectroscopy Studies on Magneto-Optical Media and Plastic Substrate Interface," IEEE Trans. Magnetics, vol. 2, pp. 740-741, Aug. 1987. (IEEE Transactions)
12. S.P. Bingulac, "On the Compatibility of Adaptive Controllers," Proc. Fourth Ann. Allerton Conf. Circuits and Systems Theory, pp. 8-16, 1994. (Conference proceedings)
13. J. MacQueen, "Some Methods for Classification Analysis of Multivariate Observations," Proc. Fifth Berkeley Symp. Math. Statistics and Probability, pp. 281-297, 1967. (Conference proceedings).
14. J. Williams, "Narrow-Band Analyzer," PhD dissertation, Dept. of Electrical Eng., Harvard Univ., Cambridge, Mass., 1993. (Thesis or dissertation)
15. E.E. Reber, R.L. Michell, and C.J. Carter, "Oxygen Absorption in the Earth's Atmosphere," Technical Report TR-0200 (420-46)-3, Aerospace Corp., Los Angeles, Calif., Nov. 1988. (Technical report with report number)
16. L. Hubert and P. Arabie, "Comparing Partitions," J. Classification, vol. 2, no. 4, pp. 193-218, Apr. 1985. (Journal or magazine citation),
17. R.J. Vidmar, "On the Use of Atmospheric Plasmas as Electromagnetic Reflectors," IEEE Trans. Plasma Science, vol. 21, no. 3, pp. 876-880, available at http://www.halcyon.com/pub/journals/21ps03-vidmar, Aug. 1992. (URL for Transaction, journal, or magazine).

Календер Мурат. Программист компании InfoTech (г. Стамбул, Турция). Окончил Университет «Едитепе» (Стамбул, Турция) в 2007 году. Бакалавр наук по вычислительной технике. Специалист в области компьютерной графики, обработки изображений и компьютерного зрения. E-Mail: mkalendertr@yahoo.com.

Панфилов Петр Борисович. Профессор кафедры вычислительных систем и сетей МИЭМ. Окончил МИЭМ в 1982 году. Кандидат технических наук, доцент. Автор свыше 50 научных работ. Специалист в области систем имитационного моделирования реального времени, интерактивной компьютерной графики и виртуальной реальности, человеко-машинных интерфейсов. E-Mail: panfilov@miem.edu.ru.