Тестирование систем реалистичной компьютерной графики

А.Г. Волобой, Е.Ю. Денисов
Институт прикладной математики им. М.В.Келдыша РАН, Москва

АННОТАЦИЯ

В работе описаны технологии тестирования программных комплексов компьютерной графики, разрабатываемых в ИПМ им. М.В. Келдыша РАН. Сложность разрабатываемых продуктов, необходимость выпуска большого числа версий (50-70 версий в год) потребовали разработки собственной системы тестирования, позволяющей эффективно выявлять ошибки и исправлять их до выпуска продукта. Тестирование продуктов компьютерной графики имеют свою специфику, на которой и сделан акцент в работе.

Ключевые слова: *системы реалистичной визуализации, тестирование программных комплексов, регрессионное тестирование, сравнение изображений*

1. ВВЕДЕНИЕ

Тестирование программного обеспечения, особенно программных продуктов, поставляемых дизайнерам, инженерам и проектировщикам, всегда являлось важной и сложной проблемой. Современные программные комплексы синтеза фотореалистичных изображений, основанные на моделировании физических законов распространения света, достигли такого уровня сложности, на котором классическое тестирование этих комплексов становится неэффективным, а часто и невозможным по ряду причин.

Наряду с классическими подходами к тестированию, заложенными еще Майерсом [1], тестирование таких комплексов имеет также свою специфику. Отметим два наиболее сложных аспекта этого тестирования:

- Проверка правильности моделирования физических явлений;
- Интерактивность программных комплексов.

Системы реалистичной визуализации стремятся к все большей достоверности генерируемых изображений. Для этого некоторые системы не только предлагают спектральное моделирование, но и поддерживают такие сложные явления как поляризация и флюоресценция света. Верификация соответствующих алгоритмов становиться самостоятельной непростой задачей. Одна из попыток серьезного подхода к этому вопросу опубликована в [2]. В статье предложены как аналитические тесты, проверяющие правильность выполнения законов оптики, так и комплексные визуальные тесты, сравнивающие полученные итоговые изображения с фотографией.

Сложность тестирования интерактивных комплексов компьютерной графики состоит в их автоматизации. Авторами были протестированы десятки программных пакетов, позволяющих в том или ином виде записать и воспроизвести действия пользователя в интерактивной программе. Среди них: AutoIt [3], Test Automation FX [4], TestComplete [5] и другие. К сожалению, обнаруженные недостатки не позволили полноценно использовать для тестирования создаваемых программных комплексов ни один из этих пакетов. Среди этих недостатков: запись действий пользователя в абсолютных координатах, что не позволяет воспроизводить записанную сессию на другом компьютере с другим разрешением экрана; неспособность распознать UI элементы пакета Qt, который используется нами при создании программного продукта; невозможность записи сессии в реальном времени (сценарий должен быть предварительно написан на специальном языке); отсутствие поддержки новейших версий Windows, а также ошибки в работе (иногда приводящие к сбою всей OS). Такая ситуация потребовала создания собственного средства записи и воспроизведения действий пользователя.

При разработке собственной системы тестирования программных продуктов компьютерной графики можно выделить следующие дополнительные особенности, присущие нашим продуктам:

- Много разнородных комплексов имеют в своей основе общие программные компоненты (например, ядро оптического моделирования);
- Частые выпуски новых версий;
- Ограниченные ресурсы и время.

В частности, предлагаемые решения применяются для тестирования набора из десяти программных комплексов; частота выпуска новых версий – 50-70 версий в год; количество программных компонент – более 500; коллектив разработчиков состоит из 15 человек.

В таких условиях абсолютно необходимым является комплексный подход к автоматизации тестирования. Предложенные и описанные далее методы позволяют добиться приемлемого качества тестирования, а значит и приемлемого качества выпускаемых программных комплексов.

Автоматизированному тестированию подвергаются как отдельные программные компоненты (низкоуровневое тестирование), так и весь программный продукт в целом (высокоуровневое тестирование). В основном используется регрессионное тестирование, проверяющее совпадение поведения новой реализации системы и её предыдущей реализации. За исключением намеренно внесенных изменений и модификаций алгоритмов результаты моделирования двух версий программного комплекса должны совпадать.

2. РЕГРЕССИОННОЕ ТЕСТИРОВАНИЕ

2.1 Покомпонентное тестирование

После каждого изменения программной компоненты кроме непосредственного содержательного тестирования вновь добавленной функциональности проводится также её регрессионное автоматическое тестирование. Результат вычисления сравнивается с ожидаемым, и выдается заключение "тест пройден - не пройден". В последнем случае разработчиком исследуется ожидаемый и реальный вывод и производится исправление выявленных ошибок.

Для обеспечения автоматизации тестирования потребовалось создать несколько систем для сравнения результатов. Поскольку результатами работы различных

программных компонент являются как текстовые данные, так и изображения, потребовалось создать подсистемы сравнения и анализа для каждого из этих типов данных.

На рис. 1 приведён пример автоматического регрессионного тестирования компоненты аппаратно-независимой реализации математических библиотек; в процессе тестирования проверяются как точность реализации математических функций, так и скорость работы данной компоненты.

```
--- Run math\test\bin\math_test.exe
***Testing SSE version of Cos on [0,PI/2]***
The precision is 1.35103e-007
The acceleration is 14.401666
***Testing SSE version of Cos on [0, PI]***
The precision is 5.36442e-007
The acceleration is 6.695166
***Testing SSE version of Sin on [0, PI]***
The precision is 1.02073e-006
The acceleration is 7.321569
***Testing SSE version of Arcsin***
The precision is 6.76808e-005
The acceleration is 17.923254
***Testing SSE version of Arccos***
The precision is 6.77109e-005
The acceleration is 16.912662
***Testing the Binary search...***
The acceleration is 2.564422
    SUCCESS!
--- Run math\perf\test\bin\math_perf.exe
   Number of iterations 262144000
```

Рис. 1. Пример автоматического тестирования математической компоненты.

Подсистема сравнения изображений обеспечивает качественный и количественный анализ различий в сравниваемых изображениях. Ниже приведен пример работы этой подсистемы (рис. 2): сравниваемое изображение показано слева, разница изображений, полученных разными версиями, в графическом представлении для последующей оценки разработчиком приведена на рисунке справа. Кроме этого для каждого сравнения сохраняются количественные данные: абсолютные и относительные ошибки в различных цветовых пространствах, гистограмма ошибочных пикселей и т.д. Эти данные могут быть проанализированы разработчиком при необходимости.

Рис. 2. Пример работы системы сравнения изображений.

Важной особенностью является то, что сравнение изображений происходит в автоматическом пакетном режиме. Допустимый порог погрешности (разница в значения несовпадающих пикселей и их допустимое количество) задается предварительно, и при превышении этого порога тестируемая подсистема считается ошибочной и подлежащей дальнейшему анализу и исправлению ошибок. Весь процесс тестирования автоматически протоколируется в журнал, содержащий список проверенных модулей и результаты их тестирования.

2.2 Тестирование готового программного продукта

Помимо покомпонентного низкоуровневого тестирования готовый программный продукт проходит также высокоуровневое тестирование, обеспечивающее проверку качества совместной работы компонент и их правильного использования разработчиком. Такое тестирование проводится также в пакетном режиме с использованием специальных модулей пакетного выполнения, являющихся частью программного комплекса, либо с использованием скриптов (в тех комплексах, которые поддерживают их выполнение).

Для тестирования интерактивных компонент, т.е. элементов интерфейса, требующих взаимодействия с пользователем, система тестирования предоставляет разработчику возможность проверить функционирование компоненты в интерактивном режиме. На рис. 3 показана подсистема отображения графиков, которая проходит автономное тестирование.

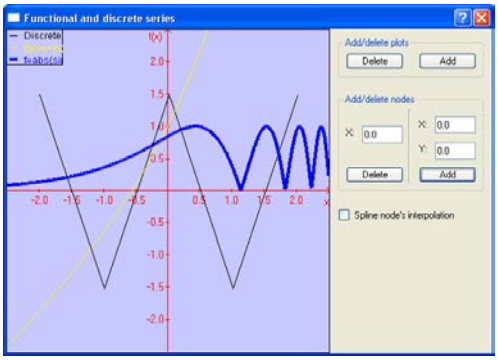

Рис. 3. Автономное тестирование компоненты.

Реально протестировать все возможные варианты использования комплекса компьютерной графики невозможно. Из-за огромного числа всевозможных опций и методов построения изображения время такого тестирования стремиться к бесконечности. При этом тестирование версии должно быть произведено в разумное время. Отсюда возникает задача создания таких тестовых данных, чтобы за одно моделирование и генерацию изображения было проверено как можно большее число различных методов или их опций. На рис. 4 представлен пример одновременного тестирования нескольких функциональностей программного продукта. В данной специально созданной тестовой сцене проверяется правильность функционирования различных алгоритмов наложения текстур на различные типы и формы поверхностей. Для анализа правильности результата вновь используется подсистема сравнения изображений, описанная выше.

Предложенный подход помимо полной автоматизации тестирования позволяет производить тестирование на нескольких компьютерах параллельно, тем самым позволяя существенно сократить время, затрачиваемое на него. Используется как так называемое "натуральное распараллеливание", когда разные компоненты тестируются на разных компьютерах, так и параллельное вычисление в самих программных комплексах, которое дополнительно тестирует и функциональность параллельных вычислений.

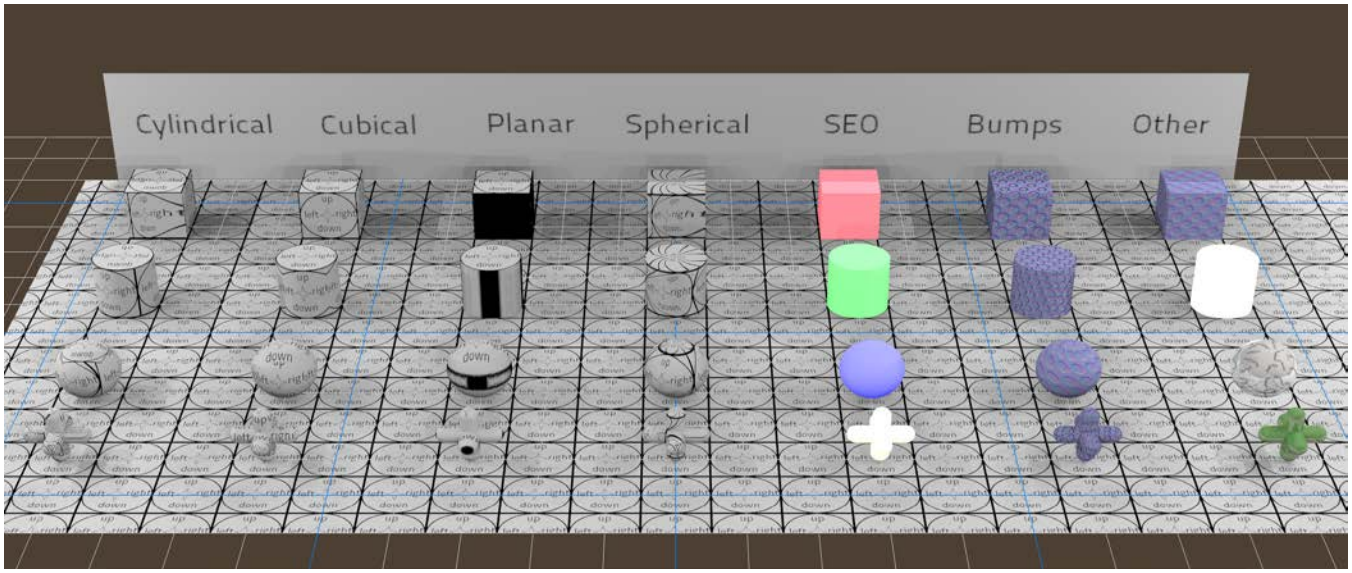

Рис. 4. Одновременная проверка различных алгоритмов

3. АВТОМАТИЧЕСКОЕ ТЕСТИРОВАНИЕ ИНТЕРАКТИВНЫХ ДЕЙСТВИЙ

Разработанная нами система записи и воспроизведения интерактивных действий пользователя названа TIPE – Testing Interactive Program Execution. Её функциональность можно сравнить с системами записи и воспроизведения макро команд, доступными в некоторых редакторах, однако возможности, реализуемые TIPE, гораздо шире.

3.1 Базовые принципы TIPE

Все действия, производимые пользователем в программном комплексе, перехватываются и записываются модулем TIPE в файл. Позднее этот файл может быть воспроизведен, в том числе на другом компьютере, отличном по характеристикам от компьютера, использованного при записи. Выполнение записанных действий в программном комплексе произойдет в том же порядке, в каком оно было записано.

Система TIPE производит запись событий, которые изменяют состояние приложения, и реакцию приложения на эти события. Приложение рассматривается как конечный автомат. Записываемые и воспроизводимые события являются входными данными для этого автомата. Записываются не только события, непосредственно выполняемые пользователем (движения мыши, нажатия клавиш на мыши и на клавиатуре), но и другие события, приводящие к смене состояний приложения: сигналы программных таймеров, системные сообщения OS и прочие. Сам модуль TIPE и его использование являются “прозрачными” для программиста – для использования функциональности TIPE в процессе программирования не нужно прилагать никаких специальных усилий. Модуль TIPE автоматически перехватывает значимые события и реакцию приложения и производит их запись.

Для успешного функционирования системы TIPE необходимо выполнение следующих условий:

- Приложение при записи и воспроизведении должно быть тем же самым;
- Начальное состояние приложения должно совпадать;
- Изменения состояний приложения, вызванные одинаковыми действиями, должны быть одинаковыми (в частности, приложения, использующие генераторы псевдо-случайных чисел, должны иметь возможность повторять ранее созданную последовательность случайных чисел);
- Все события, приводящие к изменению состояния приложения, должны записываться TIPE.

При нарушении одного из условий воспроизведение записанной сессии невозможно. Такими нарушениями могут быть: различное начальное состояние приложения (например, отличаются загруженные данные); различное поведение приложения в ответ на одинаковые действия; изменение состояний приложения зависит от оборудования; критические ситуации в процессе воспроизведения, отсутствовавшие в процессе записи (например, нехватка дискового пространства).

Кроме как средство регрессионного тестирования системе TIPE используется также:

- Для демонстрации возможностей приложения;
- В качестве интерактивного дополнения к инструкции по эксплуатации приложения;
- Средство для визуального представления действий пользователя (например, в процессе обнаружения и исправления ошибок в приложении).

3.2 Основные свойства

Основными свойствами системы TIPE являются:

Надежное воспроизведение записанных действий – гарантируется, что при соблюдении вышеуказанных условий записанная сессия работы приложения может быть воспроизведена (независимо от того, на каких компьютерах производится запись и воспроизведение). При изменении версии приложения по возможности обеспечивается обратная совместимость: если новые функциональности не влияют на записанную последовательность изменения состояний приложения, то новая версия приложения будет успешно воспроизводить сессию TIPE, записанную на предыдущей версии приложения.

Проверка условий воспроизведения – при старте воспроизведения система проверяет соответствие

нескольких геометрических параметров (таких как разрешение экрана, размеры шрифтов, размеры окон).
Проверка совпадения последовательности событий – в процессе воспроизведения система проверяет совпадение записанных событий с событиями, генерируемыми приложением при изменении его состояния.
Операции с мышью и клавиатурой визуализируются – движения мыши визуализируются перемещением курсора, нажатия клавиш мыши и клавиатуры отображаются на экране в виде всплывающих сообщений.
Имеется возможность добавления комментариев – в процессе записи сессии возможно в любое время вставить текстовый комментарий, который будет показан во всплывающем окне в процессе воспроизведения. Комментарии могут быть изменены, удалены или вставлены так же после окончания записи, путем исправления файла, содержащего записанную сессию TIPE.

При невозможности дальнейшего воспроизведения сессии (например, несовпадение начальных условий или несовпадение реакции приложения с ожидаемой реакцией) воспроизведение останавливается, и выводится сообщение об ошибке. Кроме того, в процессе записи и воспроизведения создается лог-файл, содержащий все сообщения об ошибках.

Скорость воспроизведения регулируется в широких пределах, в любой момент в процессе записи и воспроизведения процесс может быть временно приостановлен, или полностью прекращен.

Сессия TIPE хранится в файле текстового формата, что допускает внесение изменений без перезаписи всей сессии, путем изменения, добавления или удаления частей сессии. Файл содержит идентификационную информацию: название, автор, дата записи, наименование и версия приложения, другая текстовая информация. Файл имеет малый размер, что допускает его пересылку любым электронным способом.

Возможна также запись новой сессии в процессе воспроизведения имеющейся. В случае остановки воспроизведения запись новой сессии может быть продолжена по другому сценарию.

Запись сессии приложения производится, даже если явной команды записи нет. Это позволяет в случае ошибки программного комплекса в дальнейшем повторить действия, приведшие к ней.

4. РЕЗУЛЬТАТЫ

Разработанная и описанная в работе система тестирования постоянно используется при создании и поддержки нескольких систем синтеза реалистичных изображений и оптического моделирования, в частности, для систем, представленных в [6-7]. Использование описанного подхода позволило ускорить разработку и реализацию этих систем и одновременно снизить трудозатраты на их создание, тестирование и сопровождение.

Приведем характерные времена, затрачиваемые на тестирование программных продуктов. Автономное тестирование программных компонент проводится 1-2 раза в месяц и требует порядка суток компьютерного времени. Регрессивное тестирование продукта проводится при каждом выпуске (поддерживается несколько продуктов, поэтому частота – 2-4 раза в месяц) и занимает 2 компьютера в течение 2 суток. Тестирование пользовательского интерфейса (TIPE) проводится в среднем раз в три месяца и требует загрузки 2 компьютеров в течение 2 суток.

Безусловно, важным аспектом успешного тестирования является эффективный набор тестовых сцен. В идеале каждая сцена, приведшая к ошибке, после ее исправления должна быть включена в набор тестов. Иногда это возможно. Но при включении в тестирование всех ошибочных сцен время тестирования многократно возрастает, и такой подход становится неприменим. Поэтому разработка эффективных тестовых данных является сложной и творческой задачей.

Работа поддержана грантами РФФИ № 11-01-00870, 12-01-00560, а также фирмой Integra Inc. (Япония).

5. ЛИТЕРАТУРА

[1] Г. Майерс, Т. Баджетт, К. Сандлер Искусство тестирования программ, 3-е издание — М.: «Диалектика», 2012. — 272 с.

[2] В.А. Дебелов, Д.С. Козлов. Верификация алгоритмов фотореалистического рендеринга кристаллов // Труды 20-й Международной Конференции по Компьютерной Графике и Зрению, Санкт-Петербург, 2010, с.238-245.

[3] Autoit automation and scripting language, http://www.autoitscript.com

[4] Test Automation FX, corporate website, http://www.testautomationfx.com.

[5] SmartBear Software, corporate website, http://smartbear.com/products/qa-tools/automated-testing-tools.

[6] Д.Д. Жданов, И.С. Потемин, В.А. Галактионов, Б.Х. Барладян, К.А. Востряков, Л.З. Шапиро. Спектральная трассировка лучей в задачах построения фотореалистичных изображений // "Программирование", № 5, 2011, с. 13-26.

[7] A. Ignatenko, B. Barladian, K. Dmitriev, S. Ershov, V. Galaktionov, I. Valiev, A. Voloboy. A Real-Time 3D Rendering System with BRDF Materials and Natural Lighting // Proc. 14th International Conference on Computer Graphics and Vision GraphiCon-2004, Russia, Moscow, September 6 -10, 2004, p. 159-162.

Abstract

This paper describes the testing technology of computer graphics software systems being developed in the Keldysh Institute of Applied Math RAS. The complexity of products in development, the need to produce a large number of versions (50-70 versions per year) required to develop its own testing system, which effectively allows identifying errors and correcting them before the release of the product. Testing of computer graphics software has its own specifics. These specifics are addressed in the paper.

Keywords: *realistic rendering software, software system testing, regression testing, comparison of images*

Authors:

Alexey G. Voloboy, PhD, senior researcher, KIAM RAS, E-mail: voloboy@gin.keldysh.ru

Eugeny Yu. Denisov, researcher, KIAM RAS.

E-mail: eed@gin.keldysh.ru