Проблемы автоматизации тестирования программной реализации OpenGL SC

Барладян Б.Х., Шапиро Л.З., Хлупина А.А., ИПМ им. М.В. Келдыша РАН
obb@gin.keldysh.ru pls@gin.keldysh.ru nastyak@gin.keldysh.ru

Аннотация

Рассмотрены различные подходы к автоматизации тестирования программной реализации библиотеки OpenGL SC предназначенной для использования в авиационных встраиваемых системах.

1 Введение

В работе [1] сформулированы требования к операционной системе реального времени, предназначенной для работы с интегрированной модульной авионикой. В частности, должна быть обеспечена поддержка стандартов OpenGL SC/ES [2], ARINC 653, 661 [3] и возможность сертификации по стандарту DO-178 [4]. Необходимость сертификации требует соблюдения корректных процессов разработки программного обеспечения, а также полного доступа к исходным кодам библиотеки OpenGL. Важным требованием является независимость от платформы – ядро библиотеки не должно содержать кода, специфичного для конкретной архитектуры или аппаратной платформы

В работе [5] предложены методы повышения эффективности программной реализации библиотеки OpenGL SC для ее использования в авиационных встраиваемых системах. Алгоритмы растеризации были оптимизированы с учетом специфики авиационных приложений. Однако в соответствии со стандартами ARINC [3] и DO-178 [4] критически важным является обеспечение контроля качества разработанной библиотеки на всех этапах ее жизненного цикла. Для этого необходимо разработать методы эффективного тестирования, как ядра библиотеки, так и ее реализации на различных платформах. Желательно, что бы тестирование было бы максимально автоматизировано и покрывало набор функциональностей библиотеки максимально полно. Наиболее естественным подходом является использование специальных тестовых приложений, максимально покрывающих весь набор используемых функциональностей и результатом работы каждого из которых является одно изображение. Сравнивая полученные изображения с эталонными изображениями можно делать выводы о прохождении тестов и обнаруживать проблемные места. Ясно, что тесты должны быть комплексными, то есть покрывать максимальное количество функций библиотеки и их сочетаний. С другой стороны, приложения, используемые в авиационной промышленности, являются достаточно специфическими и используют достаточно ограниченный набор функций библиотеки и их сочетаний. Более того требования к графическому интерфейсу в кабине пилотов стандартизированы в стандарте ARINC 661. В связи с этим в качестве набора тестовых приложений желательно иметь такие, которые синтезируют именно те изображения, которые генерируются в реальных приложениях и, соответственно, используют соответствующие сочетания команд OpenGL.

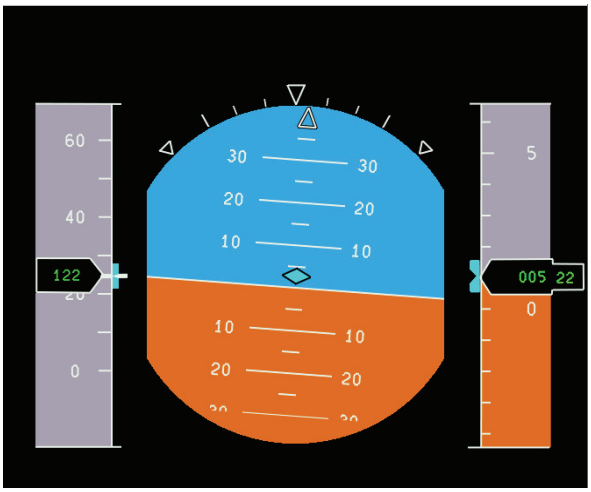

Рис. 1. Упрощенный вариант пилотажно-навигационного дисплея.

Широкий класс авиационных приложений используется для вывода на экран дисплея различной пилотажно-навигационной информации в форме удобной для восприятия пилотами. Следует отметить, что эти приложения, как правило, используют двумерные геометрические объекты для визуализации различной пилотажно-навигационной информации – скорости, высоты полета, углов поворота, различной текстовой информации и других данных. Это существенно ограничивает ко-

личество используемых функций библиотеки OpenGL. Примеры изображений, синтезируемых такими приложениями, приведены на рисунках 1 – 4.

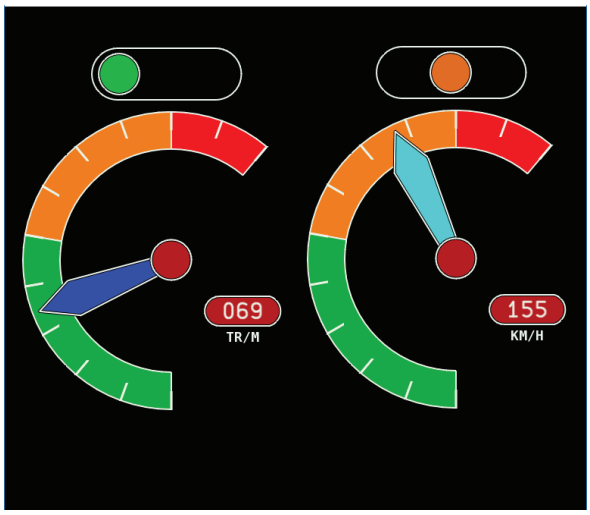

Рис. 2. Вывод показаний датчиков скорости в цифровой и аналоговой форме, с использованием цвета для указания пилоту диапазона, в котором находится значение контролируемой величины в данный момент.

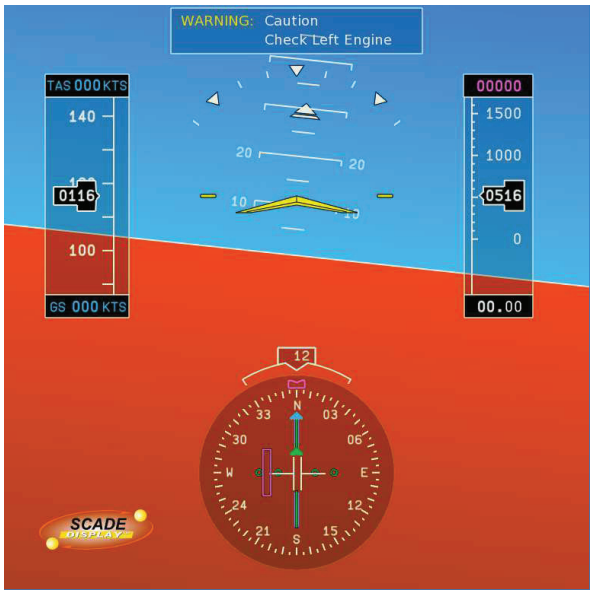

Рис. 3. Более полный вариант пилотажно-навигационного дисплея

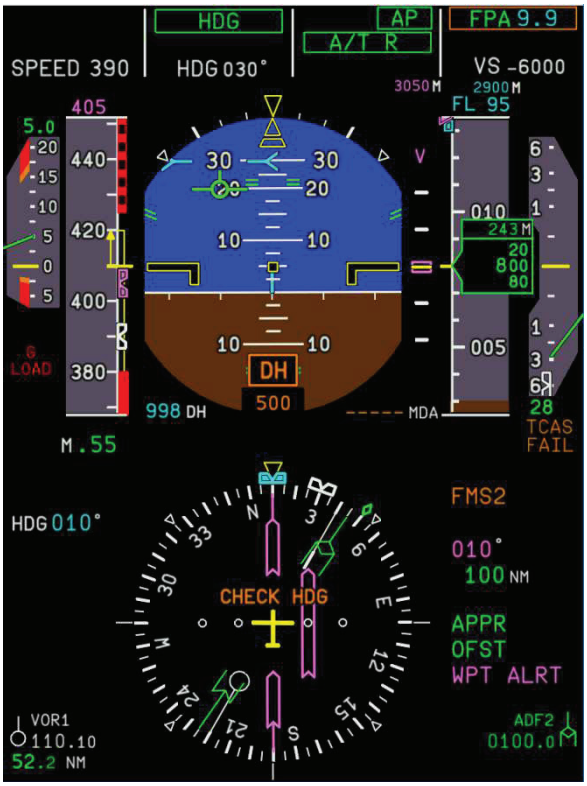

Рис. 4. Комплексный вариант пилотажно-навигационного дисплея

2 Система автоматизированного тестирования

Вывод на дисплей в стандарте OpenGL не определен. Его обеспечивают ‘нижележащие’ графические системы, такие как Win32 GDI, или X11. Промышленный консорциум Khronos Group разработал открытый стандарт интерфейса EGL [6] между OpenGL и базовыми платформами, однако взаимодействие между библиотеками OpenGL и EGL скрыто для пользователей. Использование существующей библиотеки EGL для разрабатываемой библиотеки OpenGL невозможно без взаимодействия с с разработчиками EGL и аппаратуры. Однако при тестировании ядра и реализаций OpenGL на различных платформах большая часть функциональностей может быть проверена путем сравнения синтезированных изображений с эталонными. Для тестирования ядра OpenGL необходимо сравнивать матрицы цветов пикселей, которые потом будут отображены на дисплее.

Регрессионные тесты должны сравнивать попиксельно определенное количество пар изображений. Каждая пара относится к определенному кадру определенного приложения и состоит из эталонного изображения, сделанного при помощи эталонной библиотеки

OpenGL, и тестируемого, сделанного тестируемой библиотекой.

Для создания системы автоматизированного тестирования, таким образом, следует выполнить следующие необходимые предварительные этапы:

1. Для каждого из реальных приложений выбрать необходимые для тестирования кадры.
2. Создать тестовые приложения для получения эталонных и тестируемых изображений для выбранных кадров.
3. Создать эталонные изображения для выбранных для тестирования кадров.
4. Определить правила (критерии) сравнения изображений.

3 Выбор кадров для тестирования

Первым предварительным этапом необходимым и независящим от последующих действий является выбор для каждого реального приложения кадров, на которых будут проводиться регрессионные тесты. Основное требование к выбору набора кадров это максимально возможное покрытие множества команд OpenGL и их комбинаций, используемых в реальных приложениях на столько, на сколько это возможно. Возможны три подхода к выбору тестовых кадров:

1. Выбор тестируемых кадров непосредственно разработчиками приложений, которые хорошо представляет содержательные отличия различных кадров с точки зрения использования OpenGL;
2. Выбор тестируемых кадров из визуализируемой последовательности через равные промежутки времени;
3. Случайный выбор тестируемых кадров из визуализируемой последовательности;

Следует отметить, что реальные авиационные приложения, как правило, выводят на экран дисплея информацию с различных датчиков и, следовательно, работать с ним автономно нельзя. Поэтому для отладки и тестирования этих приложений создают специальные тестовые приложения, которые эмулируют работу этих датчиков автономно. Одним из способов создания таких приложений является использование средств пакета ANSYS Scade Suite [7]. В дальнейшем под термином реальное приложение мы будем понимать именно такое автономное тестовое приложение, моделирующее работу реального приложения, работающего с различными датчиками летательного аппарата.

Наиболее корректным, с первого взгляда, представляется первый подход для выбора тестовых кадров, поскольку разработчик приложения должен лучше других представлять себе используемые функциональности библиотеки OpenGL. Однако на практике это оказывается не совсем так, поскольку разработчики используют библиотеку OpenGL не на прямую. Авиационные приложения в настоящее время, как правило, создаются с помощью инструментов пакета SCADE Display [8], где использование конкретных вызовов библиотеку OpenGL в некоторой степени скрыто от пользователей. Кроме того желательно накопить некоторое количество специальных тестовых приложений для ядра библиотеки OpenGL, которые покрывают большую часть используемых в настоящее время функциональностей. Этот набор тестов должен использоваться каждый раз, когда вносятся изменения в библиотеку, при переносе ее на новую платформу, изменении компилятора или его опций и т. д. Таким образм мы приходим к выводу, что все три, описанных выше подхода, могут использоваться при выборе кадров для тестирования. Представляется также разумным использование подхода со случайным выбором кадров для тестирования реальных приложений путем сравнения получаемых изображений на данном оборудовании, компиляторе и операционной системой с изображениями, получаемыми на эталонной системе.

4 Создание тестовых приложений

Тестовое приложение должно создавать эталонное и тестируемое изображения для заданного кадра заданного реального приложения, обеспечивая вывод его в файл без потери информации.

Итак, нужно обеспечить, чтобы приложение генерировало только нужную последовательность кадров. Из проведенного анализа приложений было установлено, что параметры OpenGL для каждого кадра переустанавливаются полностью, т.е. нет перехода каких-то из них от кадра к кадру. Кадр от кадра отличаются разными наборами параметров пилотажно-навигационных данных, по. которым и строится изображение. Поэтому надо на уровне реального приложения обеспечить соответствующее изменение параметров от

кадра к кадру, и проброс создания изображений (обращений к OpenGL) для ненужных кадров. Таким образом, мы получим из реального приложения предварительное тестовое приложение, которое обрабатывает только нужные кадры.

Далее можно рассмотреть два подхода для создания тестовых приложений:

1. Использование в качестве тестовых приложений непосредственно описанное выше предварительное тестовое приложение.
2. Изготовление на базе предварительных тестовых приложений специальных тестовых приложения, синтезирующие такие же изображения для данных кадров, что и реальные приложения

Для оценки первого подхода следует отметить, что реальные приложения, создаются, как правило, с помощью инструментов пакета SCADE Display [8]. Поэтому они достаточно громоздкие. Это их свойство, которое перейдет к предварительным тестовым приложениям, является существенным недостатком первого подхода. Полученные тестовые приложения при этом будут слишком сильно привязаны к исходным приложениям и используемых в них библиотек и средств компиляции. К достоинствам этого подхода следует отнести то, что приложение достаточно просто переделывается в тестовое. Необходимо только обеспечить вывод изображений в файл и вывод нужных кадров. Схема реализации этого подхода приведена на рисунке 5.

Рис. 5 Схема использование реального приложения как тестового

Для реализации второго подхода предлагается использование инструментов пакета Apitracer [9] или аналогичного, специально для этого созданного. Инструменты пакета Apitracer позволяют проводить трассировку вызовов команд OpenGL. Результатом трассировки является двоичный файл, содержащий последовательность всех вызовов OpenGL, т.е. всех функций со всеми параметрами, включая массивы данных и текстуры. Преимуществом использования программы Apitracer, является то, что, обрабатывая сгенерированную ей информацию, можно создать тестовое приложение, состоящее только из последовательных вызовов команд OpenGL и сопутствующей им информации (параметров функций). После этого данное тестовое приложение можно запустить с эталонной библиотеки OpenGL, и получить набор эталонных изображений. А также запустить с тестируемой библиотекой OpenGL и получить набор изображений для теста. Построенное таким образом тестовое приложение будет содержать только последовательные вызовы команд OpenGL и, следовательно, сможет использоваться на различных платформах с минимальными модификациями. Фактически мы получаем набор относительно простых тестовых приложений уже не привязанных жестко к исходным реальным приложениям. Необходимое покрытие функциональностей, используемых в исходных реальных приложениях, должно обеспечиваться технологией создания этих тестовых приложений.

Такой же подход можно реализовать, используя и разрабатываемую библиотеку OpenGL. Для этого необходимо в ней обеспечить вывод в файл каждого ее вызова – имени вызываемой функции и значений ее аргументов. Однако такой подход потребует существенного изменения кода библиотеки и, всегда, будет вызывать вопросы о степени ее соответствии библиотеке, используемой в реальных приложениях. В таком тестовом приложении следует также обеспечить вывод изображения в файл без потери информации. Схема реализации этого подхода приведена на рисунке 6.

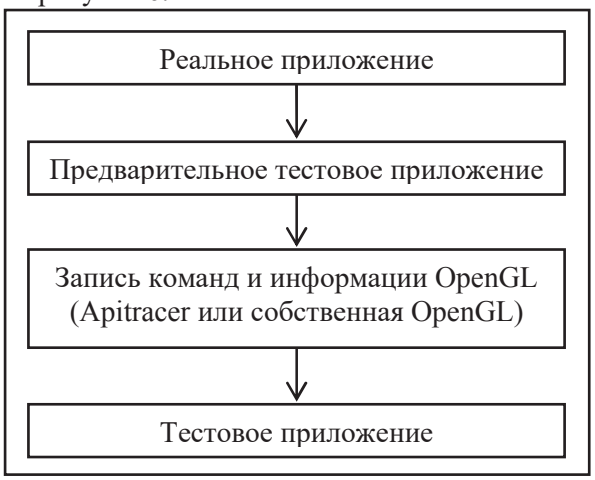

Рис. 6 Схема построения отдельного тестового приложения

Отметим, что набор тестовых приложений остаются неизменными до тех пор, пока не меняются реальные приложения, либо набор кадров для тестирования. Как правило, набор тестовых приложений может только расширяться при создании новых приложений, использующих новые возможности библиотеки OpenGL. Другая причина возможного расширения набора тестовых приложений это обнаружение каких либо проблем, не обнаруживаемых существующими тестами.

5 Создание эталонных изображений

Для создания эталонных изображений необходимо использовать все тестовые приложения с эталонной библиотекой OpenGL и с их помощью сгенерировать эталонные изображения для всех выбранных для тестирования кадров всех приложений. Эта процедура также является подготовительной. Набор полученных эталонных изображений остается неизменным до тех пор, пока не меняются реальные приложения, эталонная библиотека OpenGL, либо набор кадров для тестирования. При проведении регрессионных тестов в качестве эталонных изображений будут использоваться изображения, созданные предыдущей версией системы.

6 Сравнение изображений

Библиотека OpenGL, как правило, создает изображения в RGBA формате. Каждая компонента цвета является целым числом в диапазоне от 0 до 255 и представлена 1 байтом. Все четыре компоненты упакованы в одно четырехбайтовое целое число. В некоторых случаях может использоваться упаковка цвета пикселя в два байта. В данной работе мы не будем рассматривать этот случай. Требуемые в этом случае изменения коснутся только функции сравнения цвета пикселя.

При создании изображений реальным приложением используются следующие графические примитивы: точки, отрезки и треугольники. Для построения изображений приложение передает в используемую библиотеку OpenGL координаты точек, нормали, компоненты цветов и текстурные координаты в формате чисел с плавающей точкой (float). Для индексов вершин треугольников, и концов отрезков используются целые числа. Все операции выполняемые библиотекой при растрировании примитивов и получении изображения задаются стандартом OpenGL [2]. Поэтому несовпадение какой-то части изображений эталонной библиотеки OpenGL и тестируемой, выявленное при проведении регрессионного теста, должно рассматриваться как ошибка и как то, что библиотека не прошла тестирования. Следует также иметь в виду, что причина не прохождения теста может быть связана не только с проблемами в данной реализации OpenGL, но и с использованием нового оборудования, компилятора и его опций или новой операционной системы.

Таким образом, для проведения одного регрессионного теста необходима программа, которая побайтно сравнивает соответствующие друг другу эталонное и тестируемое изображения. Программа должна, в первом приближении, указывать индексы пикселей, для которых обнаружено несовпадения и значения цветов в этих пикселях. Для более детального анализа возникших проблем будет полезна интерактивная программа, которая должна иметь возможность показывать в графическом виде как эталонное, так и тестируемое изображения и их разность.

7 Общая схема тестирования.

Перед тестированием мы имеем набор тестовых приложений и набор эталонных изображений, которые были получены на предварительных этапах. Непосредственно тестирование разработанной библиотеки OpenGL проводится в два этапа. Сначала необходимо использовать тестовые приложения с разработанной библиотекой и сгенерировать тестируемые изображения для всех выбранных для тестирования кадров всех приложений. Затем произвести непосредственно выполнение регрессионных тестов. Схема процедуры автоматического выполнения регрессионных тестов приведена на рисунке 7.

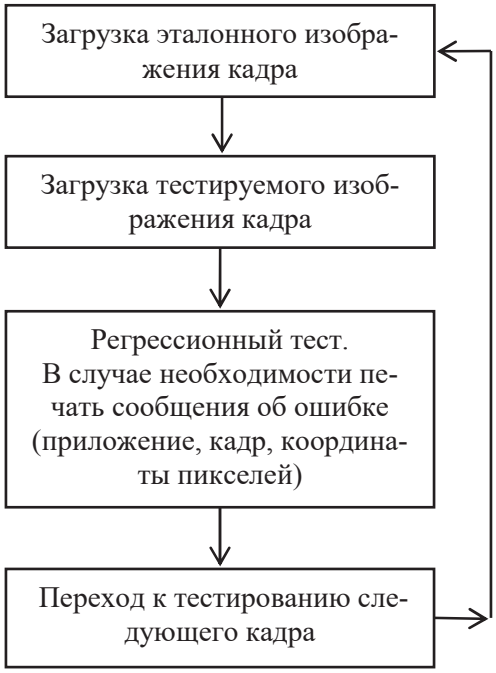

Рис. 7 Схема процедуры автоматического выполнения регрессионных тестов

8 Заключение

В работе подробно рассмотрены различные этапы работы, необходимые для построения системы автоматизированного тестирования ядра библиотеки OpenGL SC, предназначенной для использования в авиационных встраиваемых системах. Предложены различные подходы для создания тестовых приложений, в том числе и полуавтоматические. Предложенные подходы могут также использоваться для тестирования соответствующих авиационных приложений на различных платформах, при использовании различных компиляторов и операционных систем. Предложенные подходы предполагается использоваться при разработке системы автоматического тестирования библиотеки OpenGL SC, предназначенной для использования в авиационных встраиваемых системах.

Список литературы

[1] Федосов Е.А., Ковернинский И.В., Кан А.В., Солоделов Ю.А., «Применение операционных систем реального времени в интегрированной модульной авионике.OSDAY 2015, http://osday.ru/solodelov.html».

[2] «Safety Critical Working Group, https://www.khronos.org/openglsc/».

[3] «ARINC Sstandards Store :https://www.aviation-ia.com/product-categories/600-series».

[4] «DO-178C Software Considerations in Airborne Systems and Equipment Certification, https://my.rtca.org/nc__store?search=DO-178».

[5] Б.Х. Барладян, А.Г. Волобой, В.А. Галактионов, В.В. Князь, И.В. Ковернинский, Ю.А. Солоделов, В.А. Фролов, Л.З. Шапиро, «Программная реализация OpenGL SC для авиационных встраиваемых систем // Труды 27-й Международной конференции по компьютерной графике и машинному зрению GraphiCon-2017, Пермь, 24–28 сентября 2017 года, с. 47-50.».

[6] «EGL Core API Specification and Headers. // https://www.khronos.org/registry/EGL/».

[7] «ANSYSScade Suite. http://www.esterel-technologies.com/products/scade-suite/».

[8] http://www.esterel-technologies.com/products/scade-suite/.

[9] «Tools for tracing OpenGL, Direct3D, and other graphics APIs. // http://apitrace.github.io/».