DOI: 10.15514/ISPRAS-2026-38(3)-42

Многослойная визуализация в графической компоненте операционной системы реального времени

А.Г. Волобой, ORCID: 0000-0003-1252-8294 <voloboy@gin.keldysh.ru>
Б.Х. Барладян, ORCID: 0000-0002-2391-2067 <obb@gin.keldysh.ru>
Н.Б. Дерябин, ORCID: 0000-0003-1248-6047 <dek@keldysh.ru>
Л.З. Шапиро, ORCID: 0000-0002-6350-851X <pls@gin.keldysh.ru>
Е.Ю. Денисов, ORCID: 0000-0002-0614-9100 <eed@spp.keldysh.ru>
В.А. Галактионов, ORCID: 0000-0001-6460-7539 <vlgal@gin.keldysh.ru>

Институт прикладной математики им. М.В. Келдыша РАН
Россия, 125047 Москва, Миусская пл., 4, Россия.

Аннотация. Программное обеспечение комплекса бортового оборудования гражданского авиалайнера работает под управлением операционной системы реального времени (ОСРВ), частью которой является графическая компонента, формирующая изображение дисплея пилота. На него выводится интегрированная информация об окружающей обстановке и состоянии систем самолета, передаваемая от разных приложений. В работе рассматриваются вопросы обеспечения эффективной и безопасной генерации изображений от нескольких приложений, выполняемых в разных разделах ОСРВ, на один экран. Подробно разбирается подход, использующий многослойную визуализацию, позволяющий повысить эффективность с помощью аппаратной поддержки технологии VOP (Video Object Plane). Применение этого подхода потребовало разработки алгоритмов, которые должны соответствовать авиационному стандарту ARINC 661 и обеспечивать выполнение приложений в независимых разделах. Выполнение этих требований позволит сертифицировать разработанные программы и библиотеки для использования на борту гражданских авиалайнеров. Была достигнута приемлемая скорость рендеринга 25 кадров в секунду при создании слоев изображения пятью разделами.

Ключевые слова: компьютерная графика; многослойная визуализация; дисплей пилота; платформонезависимый графический интерфейс OpenGL SC; операционная система реального времени ОСРВ.

Для цитирования: Волобой А.Г., Барладян Б.Х., Дерябин Н.Б., Шапиро Л.З., Денисов Е.Ю., Галактионов В.А. Многослойная визуализация в графической компоненте операционной системы реального времени. Труды ИСП РАН, том 38, вып. 3, часть 3, 2026 г., стр. 149–160. DOI: 10.15514/ISPRAS-2026-38(3)-42.

Multi-layer visualization in the graphics component of a real-time operating system

A.G. Voloboy, ORCID: 0000-0003-1252-8294 <voloboy@gin.keldysh.ru>
B.Kh. Barladian, ORCID: 0000-0002-2391-2067 <obb@gin.keldysh.ru>
N.B. Deryabin, ORCID: 0000-0003-1248-6047 <dek@keldysh.ru>
L.Z. Shapiro, ORCID: 0000-0002-6350-851X <pls@gin.keldysh.ru>
E.Yu. Denisov, ORCID: 0000-0002-0614-9100 <eed@spp.keldysh.ru>
V.A. Galaktionov, ORCID: 0000-0001-6460-7539 <vlgal@gin.keldysh.ru>

Keldysh Institute of Applied Mathematics RAS,
4, Miusskaya sq., Moscow, 125047, Russia.

Abstract. The software of the onboard equipment complex of a civil airliner operates under the control of a real-time operating system (RTOS), part of which is a graphic component that generates the image of the pilot's display. This display indicates integrated information about the surrounding environment and the state of the aircraft's systems, transmitted from various applications. This paper provides the efficient and secure algorithm for generation of images from multiple applications running in different RTOS partitions onto a single screen. An approach using multi-layer visualization, which improves efficiency with hardware support for Video Object Plane (VOP) technology, is considered. This approach required the development of algorithms that comply with the ARINC 661 aviation standard and enable application execution in independent partitions. Meeting these requirements will allow the developed programs and libraries to be certified for use onboard civil airliners. An acceptable rendering speed of 25 frames per second was achieved, creating image layers in five partitions.

Keywords: computer graphics; multilayer visualization; pilot display; OpenGL SC; RTOS.

For citation: Voloboy A.G., Barladian B.Kh., Deryabin N.B., Shapiro L.Z., Denisov E.Yu., Galaktionov V.A. Multilayer visualization in the graphics component of a real-time operating system. Trudy ISP RAN/Proc. ISP RAS, vol. 38, issue 3, part 3, 2026, pp. 149-160 (in Russian). DOI: 10.15514/ISPRAS-2026-38(3)-42.

1. Введение

Комплексы бортового оборудования современных гражданских авиалайнеров проектируются в соответствии с идеологией «Интегрированной модульной авионики» [1], используемой, в частности, в таких лайнерах как Airbus A320, Boeing 787, Superjet 100 и МС-21. По этой идеологии нескольких функциональных приложений, обеспечивающих программную часть той или иной самолетной системы, выполняются на одном вычислительном модуле. Такой режим работы приложений обеспечивается операционной системой реального времени (ОСРВ) [2]. Комплексы бортового оборудования относятся к системам, критическим с точки зрения безопасности (safety critical). Это накладывает дополнительные ограничения на разработку графической компоненты ОСРВ. Так, для обеспечения безопасности каждое из функциональных приложений должно выполняться в отдельном разделе, чтобы возможные ошибки в одном разделе имели бы минимальное влияние на работу приложений в других разделах. Графическая библиотека должна соответствовать специальному стандарту OpenGL SC (SC – safety critical).

Интерфейс практически всех современных самолетов построен на концепции «стеклянной кабины». Она позволяет улучшить восприятие полетных данных [3] за счет объединения важной информации на одном многофункциональном дисплее, на который выводится интегрированное, легко воспринимаемое изображение окружающей обстановки и состояние систем самолета. Кроме того, становится возможным отображать разную информацию в разных сегментах полета. Таким образом, на один дисплей одновременно выводится различная информация, генерируемая многочисленными независимыми системами

управления полетом. В общем случае информация может выводиться как в различные окна на дисплее, так и в общие для различных приложений окна.

Задача многооконной визуализации из нескольких разделов в различные окна одного дисплея рассматривалась в работах [4-6]. При этом подходе изображения генерируются независимо в каждом разделе, а затем компоновщик, выполняемый в отдельном разделе, составляет из них общую картинку, используя библиотеку кадрового буфера. Финальное изображение выводится при помощи графического драйвера, предоставляющего OpenGL SC интерфейс. Типичная авиационная платформа имеет только один графический процессор, и в силу требований безопасности OpenGL с аппаратным ускорением может работать только в одном разделе ОСРВ. Одной из возможных альтернатив в этом случае может быть использование высокоэффективной программной реализации OpenGL SC [7], но она все равно будет проигрывать в скорости рендеринга.

Следует отметить, что компоновщик позволяет реализовать только многооконный режим, когда каждое приложение может выводить информацию только в свое окно. На практике, однако, бывает необходимость выводить информацию из нескольких приложений в общее окно на дисплее, когда одна информация фактически накладывается на другую. Такой тип визуализации даст возможность сделать картинку на экране дисплея более информативной, позволяя одним взглядом охватить важную информацию, что, в конечном счете, снижает нагрузку на пилота и ведет к повышению безопасности полетов. Этот подход можно реализовать через многослойную визуализацию, тем более некоторые GPU предоставляют ее аппаратную поддержку.

Возможности использования многослойной визуализации для решения различных практических и исследовательских задач рассматриваются во многих работах. Так, например, в [8] ее применяют для получения 3D изображения микрорельефа поверхности из набора 2D изображений с малым фокусным расстоянием, сделанных микроскопом. Здесь для слияния изображений использован принцип локального максимума контраста. В работе [9] многослойность позволяет визуализировать большие объёмы данных по уровням. Вместо визуализации всего объёма данных сразу в максимальном разрешении используется последовательное приближение и повышение детализации для того участка, на который сейчас смотрит пользователь. В статье рассмотрены два подхода: вейвлет-анализ и многоуровневый метод конечных элементов. Через многослойное или многоплоскостное представление сцены решается проблема отражений при изображении световых полей (light field images) [10]. Этот метод удаления отражений не только обеспечивает превосходную производительность разделения, но и поддерживает синтез новых ракурсов разделенных результатов.

В данной работе рассматриваются аспекты многослойной визуализации дисплея пилота гражданского воздушного судна, созданной под управлением отечественной перспективной ОСРВ JetOS [11]. Показаны проблемы, встретившиеся при реализации подхода, и разработанные нами решения.

2. Реализация многослойной визуализации

Аппаратная поддержка многослойной визуализации возникла благодаря разработке видео форматов MPEG-1, MPEG-2, MPEG-4 и H.264. Не вдаваясь в детали, можно сказать, что один момент времени видео состоит из набора плоскостей видео объекта (VOP, Video Object Plane). Каждая из плоскостей содержит отдельную информацию и может обрабатываться независимо от других плоскостей, а итоговый кадр получается заданной их композицией. В первую очередь технология VOP применялась для возможности сжатия видео при сохранении качества изображения [12] и для эффективного кодирования движущихся объектов [13]. Позднее она также использовалась для генерации искусственных видео [14]. Многие GPU поддерживают стандартные команды композиции плоскостей видео объекта в

итоговый кадр. Поэтому вполне разумно использование VOP технологии для нашей многослойной визуализации.

Разработанная нами реализация специального интерфейса для графического процессора обеспечивает многослойную визуализацию путем наложения друг на друга нескольких прозрачных или полупрозрачных слоев (рис. 1). Каждый слой представляет собой изображение, генерируемое одним или несколькими разделами ОСРВ как с помощью библиотеки OpenGL SC, так и прямым рисованием. Особенностью нашей реализации является работа под ОСРВ JetOS и необходимость соответствовать авиационному стандарту ARINC 661 [15]. Реализованный интерфейс обеспечивает следующие функциональности:

1) Вывод изображений в несколько слоев с созданием отдельных окон в каждом слое;
2) Управление порядком наложения и способом смешивания слоев;
3) Управление масштабированием, смещением по оси X и Y, поворотом и отражением содержимого слоя;
4) Выбор слоя и окна для синтеза изображения средствами OpenGL.

Непосредственно графический драйвер при этом работает в отдельном разделе ОСРВ.

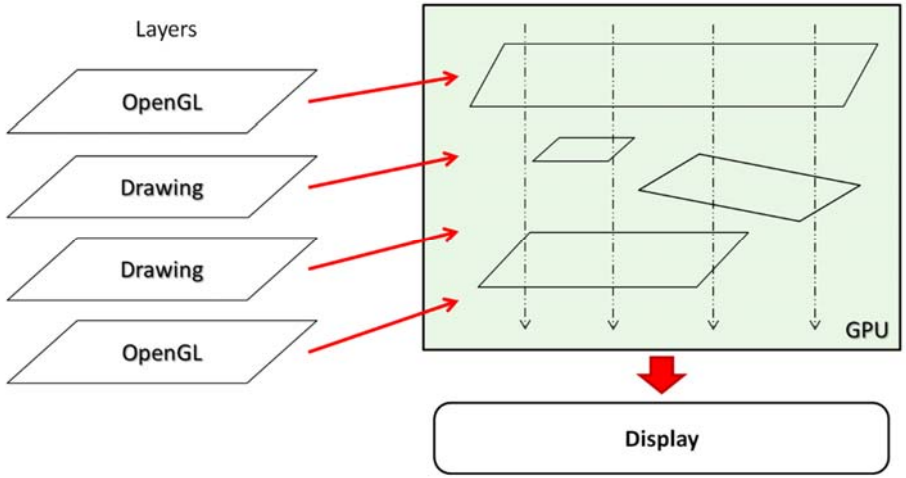

Рис. 1. Общая схема построения многослойного изображения.
Fig. 1. General scheme for constructing a multilayer image.

Слои несут смысловую нагрузку с точки зрения изображения, позволяя разделить картинки, изменяющиеся на экране с разной скоростью или в разные моменты времени. Например, один слой может быть сложным статическим изображением, создаваемым несколькими разделами, а другой – фиксированным изображением движущегося курсора или перемещающейся метки. Другое типичное применение – это наложение слоев карт, включающих рельеф, инфраструктуру, отображение погодных условий и т.д. Также эта функциональность может использоваться для вывода на экран дисплея в кабине пилота дополнительной информации, такой как текстовые сообщения или индикация критических состояний или событий.

3. Генерация слоя с использованием OpenGL

Работу приложения комплекса бортового оборудования, использующего библиотеку OpenGL для визуализации, можно разделить на две части:

5) Подготовка различных данных, необходимых для работы OpenGL – параметры геометрии, различные атрибуты и так далее;

6) Непосредственный вызов функций OpenGL SC с подготовленными параметрами.

Так как OpenGL SC может выполняться только в одном разделе ОСРВ, то подготовка данных выполняется разными приложениями в своих разделах (рисующее разделы), а вызовы функций OpenGL – в отдельном разделе (растеризующий раздел). Каждое приложение подготавливает свои данные и записывает индексы функций, необходимых для визуализации, с соответствующими параметрами в специальный буфер (массив). Он находится в общей памяти с разделом, который осуществляет реальные вызовы функций OpenGL. Когда вся информация, необходимая для генерации изображения одного кадра, подготовлена рисующими разделами, то изображение выводится на экран дисплея растеризующим разделом.

Однако приложения в рисующих разделах ничего не знают об этой схеме. Они считают, что формируют изображение с помощью библиотеки OpenGL. Для реализации этого мы разработали две библиотеки: liboglout2 для рисующих разделов и liboglin2 для растеризующего раздела. Суффикс 2 в именах этих библиотек означает, что они предназначены для работы с командами OpenGL SC стандарта 2.0 (SC2). Библиотека liboglout2 содержит набор функций, эмулирующий соответствующие функции OpenGL SC2. Вместо выполнения этих функций она записывает их идентификаторы в соответствующий участок общей памяти. В эту же общую память записываются также все параметры функций и данные, необходимые для их выполнения.

Эти эмулирующие функции имеют ту же нотификацию, что и соответствующие им функции OpenGL. Если такая функция просто передает фиксированное число параметров, то в текущую позицию записывается индекс вызываемой функции, а за ним подряд значения параметров. Это относится к большинству функций OpenGL таких, как, например, glClearColor(), glClearDepthf(), glClearStencil(), glUniform1i(). Этот же способ используется тогда, когда передаются массивы известной длины. В более сложных случаях, когда передаются указатели на массивы, длина которых не определена на момент вызова (например, при вызове glVertexAttribPointer()), технология кодирования вызова несколько усложняется. В момент вызова таких функций указатель запоминается, а данные по этому указателю используются позже при вызове функций glDrawArrays() или glDrawRangeElements(), когда необходимые размерности становятся известными. При этом массивы с запомненными указателями (как правило, координаты вершин или текстурные координаты) копируются в зарезервированную область общей памяти. Использоваться они будут в библиотеке liboglin2 при выполнении реальных функций OpenGL. Библиотека liboglout2 линкуется с рисующим разделом вместо реальной библиотеки OpenGL SC2. Сам код раздела при этом практически не изменяется. Необходимы только определенные изменения в конфигурационных файлах и некоторый код в main(), который пишется по общему шаблону. Каждый рисующий раздел создает свой слой будущего изображения.

Декодирование информации, записанной приложениями с помощью библиотеки liboglout2, в вызовы функций OpenGL осуществляется в одном растеризующем разделе последовательно для каждого пакета данных. Для этого используется библиотека liboglin2, состоящая из одной функции ProcessOGL_Input(), на вход которой подается массив, записанный в рисующем разделе. Она последовательно читает данные из массива, по прочитанному индексу функции OpenGL, переходит к участку кода, записанного для данной функции, извлекает параметры функции, включая значения указателей, и вызывает с этими параметрами заданную функцию OpenGL. Библиотека liboglin2 лннкуется с растеризующим разделом и с библиотекой, реализующей OpenGL SC2. На рис. 2 представлена схема синтеза графики из нескольких разделов с учетом использования библиотек liboglout2 и liboglin2.

Проблемы в этой схеме возникают при использовании нескольких шейдерных программ, текстур и щейдерных объектов, создаваемых в разных рисующих разделах. Идентификаторы объектов одного типа должны отличаться, быть уникальными, даже если они созданы в разных разделах. Идентификаторы назначаются или генерируются в растеризующем разделе, где происходит реальное выполнение команд OpenGL SC2. Но рисующие разделы не имеют обратной связи с растеризующим и не могут получить идентификаторы от него. Кроме того, следует учитывать, что в общем случае код приложения может быть сгенерирован разными версиями пакета SCADE [16], с помощью которого создаются программы для авионики, или написан вручную.

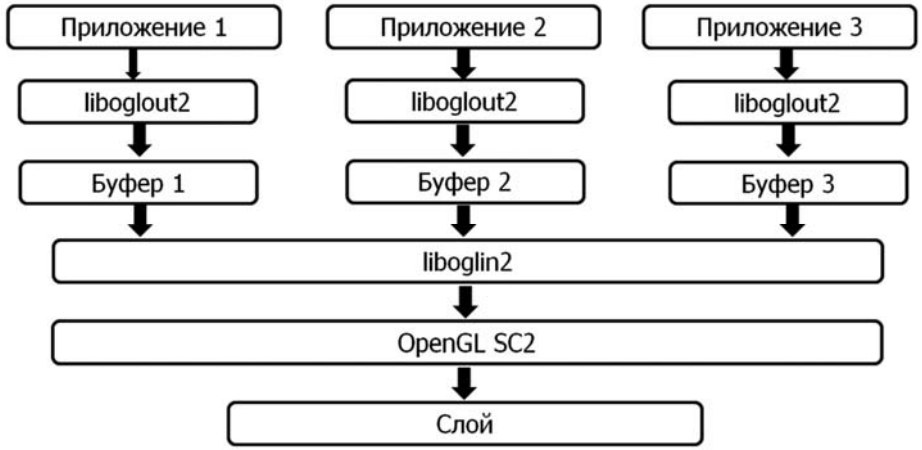

Рис. 2. Схема работы приложений с использованием библиотек liboglout2 и liboglin2.
Fig. 2. Scheme of application operation using liboglout2 and liboglin2 libraries.

Существуют четыре типа объектов, идентификаторы которых должны иметь уникальные значения:

- шейдерные программы, определяемые вызовом функции glCreateProgram();
- шейдерные объекты типа UNIFORM, определяемые вызовом функции glGetUniformLocation();
- шейдерные объекты типа VERTEX ATTRIBUTE, определяемые вызовом функции glGetAttribLocation();
- текстуры, определяемые вызовом функции glGenTextures().

Очевидно, что проблема уникальных идентификаторов должна быть решена на уровне рисующих разделов (приложений), а значит в библиотеке liboglout2. Для этой цели библиотека создает собственные внутренние определения этих идентификаторов. Экземпляры библиотеки liboglout2 в каждом разделе используют собственное адресное пространство. Поэтому для последовательной нумерации идентификаторов одного типа в разных разделах используются общие указатели идентификаторов, хранящиеся в общей памяти рисующих разделов:

```
static UINT32* prog_ident_ptr;
static UINT32* uniform_ident_ptr;
static UINT32* attrib_ident_ptr;
static UINT32* tex_ident_ptr;
```

Для связи этих указателей с библиотекой liboglout2, прилинкованной к конкретному разделу, они устанавливаются с помощью специальных функций, которые должны быть выполнены

до первого обращения раздела к функциям, эмулирующим OpenGL SC2. При вызове в рисующем разделе функции OpenGL SC2 библиотека возвращает значение по нужному указателю и увеличивает это значение на единицу. Такой подход гарантирует, что значения идентификаторов объектов одного типа при получении их из разных рисующих приложений не будут повторяться.

В библиотеке liboglin2 идентификатор объекта данного типа определяется через массив, для которого внутренний идентификатор из liboglout2 рассматривается как индекс, а значение равно идентификатору, назначенному библиотекой OpenGL SC2.

В каждом разделе теперь потенциально могут использоваться несколько собственных программных объектов. Они создаются либо явно с помощью вызова функции glCreateProgram(), либо неявно внутри библиотеки OGLX пакета SCADE. Предложенный подход в некотором смысле автоматически поддерживает оба варианта создания программных объектов. Идентификаторы текстур, объектов типа UNIFORM и типа VERTEX ATTRIBUTE используются обычным образом при прямом написании кода раздела, использующего функции OpenGL SC2. В случае использования пакета SCADE в библиотеке OGLX для идентификаторов будут автоматически использоваться внутренние переменные библиотеки liboglout2.

4. Результаты

На рис. 3 приведен результат генерации изображений из 2 разделов, каждый из которых использует 1 или 2 собственных программных объектов. В левой части экрана изображение Primary Flight Display (PFD), сгенерированное приложением, подготовленным с помощью пакета SCADE R16. В правой части экрана изображение, сгенерированное приложением «карта перемещения аэропорта» (AMM Airport Moving Map), написанного по стандарту OpenGL SC2 без использования пакета SCADE. Скорость рендеринга – 25.0 кадров в секунду. Фактически, с помощью механизма многослойной визуализации здесь реализован многооконный дисплей.

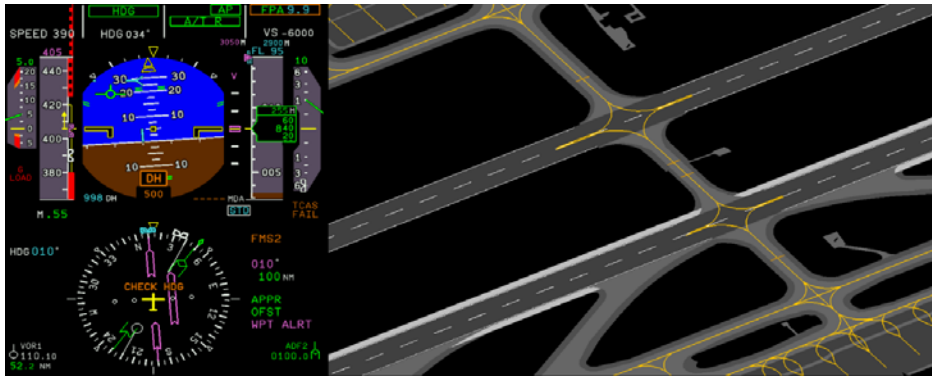

Рис. 3. Генерация изображения из 2-х слоев, каждый в своей части экрана.
Fig. 3. The image generated from 2 layers, each in its own part of the screen.

На рис. 4 мы к слоям, использованным при генерации рис. 3, добавили третий слой с информацией о географических координатах. Положение надписи из третьего слоя может перемещаться поверх изображения, как это отмечено зеленым овалом для двух разных кадрах, и варьироваться в зависимости от состояния оборудования самолета или его положения в пространстве. При этом изображения первых двух слоев и их отображение остаются неизменными.

Рис. 4. Два кадра дисплея, сгенерированные из 3-х слоев.
Один слой отвечает за перемещаемый текст.
Fig. 4. Two display frames generated from three layers. One layer is responsible for the moving text.

На рис. 5-7 приведены три снимка экрана дисплея, демонстрирующие управление слоями. На вход подается четыре слоя: PFD, DRAW, GEARS, PLANE. Два слоя – PFD и GEARS (шестеренки) – были сгенерированы приложениями с использованием OpenGL с аппаратным ускорением. Слой DRAW (самолет, меняющий цвет и размер на разных рисунках) демонстрирует возможность произвольного рисования без использования OpenGL. Слой PLANE (белый абрис самолета) демонстрирует неизменяемое изображение, перемещаемого по фиксированной траектории. На рис. 6 его белый цвет смешивается с синим цветом слоя GEAR. Слой PLANE показывает способ естественной реализации курсора.

На последнем примере демонстрируются следующие операции со слоями:

1) Изменение позиции (все слои);
2) Изменение размеров, масштабирование – слои PFD, DRAW, GEARS;
3) Изменение прозрачности – слои PFD, DRAW, GEARS.;
4) Трансформации со слоем PFD;
5) Изменение порядка видимости слоев – слой PLANE периодически меняет порядок видимости от самого верхнего до самого нижнего, при этом изменяется и порядок других слоев;
6) Альтернативный режим смешивания для альфа канала – у слоя GEARS появляется и исчезает голубой фон у шестеренок.
7) На примере, представленном на рис. 5-7, также была проверено изменение видимости. Слой DRAW периодически выключался, и изображение самолета пропадало на экране. Видно, что на рис. 5 и 6 слой GEAR расположен ближе к экрану, на рис. 7 ближе к фону. Слой PFD на рис. 7 полупрозрачный, а на двух других кадрах нет.

5. Заключение

Разработанная нами реализация специального интерфейса для графического процессора обеспечивает многослойную визуализацию путем наложения друг на друга нескольких слоев. Она основана на известной, эффективной технологии VOP – Video Object Plane. Реализация для комплекса бортового оборудования, работающего под управлением операционной системы реального времени JetOS, потребовала разработки алгоритмов,

которые должны соответствовать авиационному стандарту ARINC 661 и обеспечивать выполнение приложений в независимых разделах.

Рис. 5. Фото экрана дисплея, кадр 1. Изображение скомпоновано из четырех слоев.
Fig. 5. Photo of the display screen, frame 1. The image is composed of four layers.

Рис. 6. Фото экрана дисплея, кадр 2. Изображение скомпоновано из четырех слоев.
Fig. 6. Photo of the display screen, frame 2. The image is composed of four layers.

Рис. 7. Фото экрана дисплея, кадр 3. Изображение скомпоновано из четырех слоев.
Fig. 7. Photo of the display screen, frame 3. The image is composed of four layers.

Выполнение этих требований позволит сертифицировать разработанные программы и библиотеки для использования на борту гражданских авиалайнеров. Была достигнута приемлемая скорость рендеринга 25 кадров в секунду при создании слоев изображения пятью разделами.

Для допуска к полетам в гражданской авиации сертификации подлежит не только программный код, но и вычислительная платформа с графическим ускорителем, и сам дисплей пилота (в авиации называемый «индикатором»). Поэтому при финальном тестировании и отладке всего комплекса в качестве результата выполнения приложений рассматривались фотографии и видео экрана дисплея (рис. 5-7), а не изображения в видеопамяти (рис. 3-4). В финальной конфигурации графической системы интерфейсы, позволяющие вывести качественное изображение видеопамяти на другие носители, отсутствуют в связи с сертификационными требованиями. Сертификация комплекса бортового оборудования – это отдельная сложная задача, но на любом этапе разработки мы обязаны принимать в расчет сертифицируемость предложенных решений.

Список литературы / References

[1]. Федосов Е.А. Косьянчук В.В., Сельвесюк Н.И. Интегрированная модульная авионика. Радиоэлектронные технологии, т. 201, 2015, стр. 66-71.

[2]. Маллачиев К.М., Пакулин Н.В., Хорошилов А.В. Устройство и архитектура операционной системы реального времени. Труды ИСП РАН, 2016, т. 28, вып. 2, стр. 181-192. DOI: 10.15514/ISPRAS-2016-28(2)-12. / Mallachiev K.M., Pakulin N.V., Khoroshilov A.V. Design and architecture of real-time operating system. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 2, 2016, pp. 181-192. DOI: 10.15514/ISPRAS-2016-28(2)-12.

[3]. Kal'avsky P., Rozenberg R., Mikula B., Zgodavova Z. Pilots' Performance in Changing from Analogue to Glass Cockpits, Proc. of the 22nd Int. Scientific Conf. on Transport Means (Transport Means), 2018, pp. 1104-1109.

[4]. A Safety Critical Compositor for OpenGL SC 1.0.1 and OpenGL SC 2.0. Available at: http://www.coreavi.com/sites/default/files/compositor_whitepaper_final.pdf, accessed 30.01.2026.

[5]. EGL_EXT_compositor. Available at: http://www.coreavi.com/sites/default/files/coreavi_product_brief_-_egl_ext_compositor.pdf, accessed 30.01.2026.
[6]. Barladian B.Kh., Deryabin N.B., Shapiro L.Z., Solodelov Yu.A., Voloboy A.G., Galaktionov V.A. Multiwindow Rendering on a Cockpit Display Using Hardware Acceleration. Programming and Computer Software, 2021, vol. 47, no. 6, pp. 457-465. DOI: 10.1134/S0361768821060025.
[7]. Barladian B.Kh., Voloboy A.G., Galaktionov V.A., Knyaz' V.V., Koverninskii I.V., Solodelov Yu.A., Frolov V.A., Shapiro L.Z. Efficient Implementation of OpenGL SC for Avionics Embedded Systems. Programming and Computer Software, 2018, vol. 44, no. 4, pp. 207-212. DOI: 10.1134/S0361768818040059.
[8]. Berehulyak O.R., Vorobel R.A., Ivasenko I.B., Krechkovska H.V., Student O.Z., Nykyforchyn H.M. 3D visualization of the fracture surface by the series of multilevel images. Отбор и передача информации, 2020, вып. 48. стр. 79-85.
[9]. Ertl T., Westermann R., Grosso R. Multiresolution and hierarchical methods for the visualization of volume data. Future Generation Computer Systems, 1999, vol. 15, issue 1, pp. 31-42.
[10]. Chen J., Zhang S., Lv Y., Gao C., Lin Y. Hierarchical Interactive Multi-Plane Image Construction for Light Field Background and Reflection Separation. IEEE Transactions on Circuits and Systems for Video Technology, 2026. DOI: 10.1109/TCSVT.2026.3662736.
[11]. Солоделов Ю.А., Горелиц Н.К. Сертифицируемая бортовая операционная система реального времени JetOS для российских проектов воздушных судов. Труды ИСП РАН, 2017, том 29, вып. 3, стр. 171-178. DOI: 10.15514/ISPRAS-2017-29(3)-10. / Solodelov Yu.A., Gorelits N.K. Certifiable onboard real-time operation system JetOS for Russian aircrafts design. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 3, 2017. pp. 171-178 (in Russian). DOI: 10.15514/ISPRAS-2017-29(3)-10.
[12]. Jinzenji K., Okada S., Watanabe H., Kobayashi N. Automatic two-layer video object plane generation scheme and its application to MPEG-4 video coding. IEEE International Symposium on Circuits and Systems (ISCAS), 2000, vol. 3, pp. 606-609.
[13]. Meier T., Ngan K.N. Automatic segmentation of moving objects for video object plane generation. IEEE Transactions on Circuits and Systems for Video Technology, 1998, vol. 8, no. 5, pp. 525-538. DOI: 10.1109/76.718500.
[14]. Henderson P., Lampert C.H. Unsupervised object-centric video generation and decomposition in 3D. Advances in Neural Information Processing Systems, vol. 33, 2020. pp. 3106-3117.
[15]. Barboni E., Conversy S., Navarre D., Palanque P. Model-Based Engineering of Widgets, User Applications and Servers Compliant with ARINC 661 Specification. In: Doherty G., Blandford A. (eds) Interactive Systems. Design, Specification, and Verification. DSV-IS 2006. Vol. 4323 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2007. DOI: 10.1007/978-3-540-69554-7_3.
[16]. Ansys SCADE Suite Model-Based Development Environment for Critical Embedded Software, 2021. Available at: https://www.ansys.com/products/embedded-software/ansys-scade-suite, accessed 30.01.2026.

Информация об авторах / Information about authors

Алексей Геннадьевич ВОЛОБОЙ – доктор физико-математических наук, ведущий научный сотрудник ИПМ им. М.В. Келдыша РАН. Область научных интересов: компьютерная графика, вычислительная оптика, трассировка лучей, моделирование освещенности.

Alexey Gennadievich VOLOBOY – Dr. Sci. (Phys.-Math.), leading researcher of Keldysh Institute of Applied Mathematics of RAS. Research interests: realistic computer graphics, computational optics, ray tracing techniques, lighting simulation.

Борис Хаимович БАРЛАДЯН – кандидат технических наук, старший научный сотрудник ИПМ им. М.В. Келдыша РАН. Сфера научных интересов: компьютерная графика, OpenGL, рендеринг в реальном времени.

Boris Khaimovich BARLADIAN – Cand. Sci. (Tech.), senior researcher of Keldysh Institute of Applied Mathematics of RAS. Research interests: computer graphics, OpenGL, real-time rendering.

Николай Борисович ДЕРЯБИН – научный сотрудник ИПМ им. М.В. Келдыша РАН. Сфера научных интересов: компьютерная графика, OpenGL, операционные системы.

Nikolay Borisovich DERYABIN – researcher of Keldysh Institute of Applied Mathematics of RAS. Research interests: computer graphics, OpenGL, operation systems.

Лев Залманович ШАПИРО – кандидат технических наук, старший научный сотрудник ИПМ им. М.В. Келдыша РАН. Сфера научных интересов: компьютерная графика, OpenGL, рендеринг в реальном времени.

Lev Zalmanovich SHAPIRO – Cand. Sci. (Tech.), senior researcher of Keldysh Institute of Applied Mathematics of RAS. Research interests: computer graphics, OpenGL, real-time rendering.

Евгений Юрьевич ДЕНИСОВ – научный сотрудник ИПМ им. М.В. Келдыша РАН. Сфера научных интересов: компьютерная графика, операционные системы.

Eugeny Yurievich DENISOV – researcher of Keldysh Institute of Applied Mathematics of RAS. Research interests: computer graphics, operation systems.

Владимир Александрович ГАЛАКТИОНОВ – доктор физико-математических наук, профессор, главный научный сотрудник отдела компьютерной графики и вычислительной оптики Института прикладной математики им. М.В. Келдыша РАН. Сфера научных интересов: компьютерная графика, оптическое моделирование, компьютерная лингвистика, научная визуализация.

Vladimir Aleksandrovich GALAKTIONOV – Dr. Sci. (Phys.-Math.), Prof., Chief researcher at the Keldysh Institute of Applied Mathematics RAS. Research interests: computer graphics, optical simulation, computer linguistics, scientific visualization.