DOI: 10.15514/ISPRAS–2020–32(4)–5

Эффективные методы и алгоритмы синтеза видео 360 градусов на основе кубической проекции виртуального окружения

П.Ю. Тимохин, ORCID: 0000-0002-0718-1436 <webpismo@yahoo.de>
М.В. Михайлюк, ORCID: 0000-0002-7793-080X <mix@niisi.ras.ru>
Е.М. Вожегов, ORCID: 0000-0003-2676-1206 <vozhegovem@icloud.com>

ФНЦ Научно-исследовательский институт системных исследований РАН,
117218, Россия, г. Москва, Нахимовский просп., д. 36, к. 1

Аннотация. В статье рассматривается задача создания и воспроизведения панорамного видео с обзором 360 градусов, обеспечивающего погружение исследователя в виртуальную среду вне родительской системы виртуального окружения (СВО). Для решения этой задачи предлагается расширение метода проекции в кубическую карту, при котором разрешение карты определяется с учетом угла обзора камеры зрителя и разрешения экрана (Adequate Cubemap Projection, ACMP). В работе исследовано влияние ориентации камеры зрителя внутри куба на отношение «пиксел карты/пиксел экрана», определяющее качество визуализации панорамы, и предложен метод вычисления разрешения кубической карты для качественной визуализации панорамы при всех возможных ориентациях камеры. В работе рассмотрены эффективные метод и алгоритм создания ACMP-видео на GPU с помощью технологии рендеринга в текстуру, которые позволяют синтезировать панорамы с постоянной ориентацией или с привязкой к направлению взгляда наблюдателя. Также в исследовании предложены эффективные методы и алгоритмы воспроизведения ACMP-видео, основанные на визуализации видимых граней куба и адаптивной буферизации кадров. Полученные методы и алгоритмы реализованы в программном комплексе синтеза ACMP-видео (C++, OpenGL, FFmpeg), который включает в себя модуль захвата кадров (встраиваемый в СВО) и плеер. Разработанное решение было протестировано в системе «Виртуальная Земля» по обучению наблюдению объектов земной поверхности с орбиты Международной космической станции (МКС). С помощью модуля захвата было создано ACMP-видео полета вдоль участка подспутниковой трассы МКС. При воспроизведении данного видео обучаемый летит по орбите над виртуальной 3D поверхностью Земли и может исследовать ее, поворачивая камеру. Апробация комплекса подтвердила адекватность разработанных методов и алгоритмов поставленной задаче. Полученные научные и практические результаты позволяют расширить возможности и сферу применения СВО, систем научной визуализации, видеотренажеров и виртуальных лабораторий, эффективно обмениваться опытом между исследователями и др.

Ключевые слова: видео 360 градусов; панорамное видео; кубическая проекция; визуализация; виртуальное окружение.

Для цитирования: Тимохин П.Ю., Михайлюк М.В., Вожегов Е.М. Эффективные методы и алгоритмы синтеза видео 360 градусов на основе кубической проекции виртуального окружения. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 73–88. DOI: 10.15514/ISPRAS–2020–32(4)–5

Благодарности: Публикация выполнена в рамках государственного задания по проведению фундаментальных научных исследований (ГП 14) по теме (проекту) «34.9. Системы виртуального окружения: технологии, методы и алгоритмы математического моделирования и визуализации» (0065-2019-0012).

Efficient methods and algorithms to synthesize 360-degree video based on cubemap projection of virtual environment

P.Yu. Timokhin, ORCID: 0000-0002-0718-1436 <webpismo@yahoo.de>
M.V. Mikhaylyuk, ORCID: 0000-0002-7793-080X <mix@niisi.ras.ru>
E.M. Vozhegov, ORCID: 0000-0003-2676-1206 <vozhegovem@icloud.com>
Scientific Research Institute for System Analysis of RAS,
36, build. 1, Nakhimovskiy Avenue, Moscow, 117218, Russia

Abstract. The paper deals with the task of creation and playback of panoramic video with 360-degree overview, which allows the researcher to be immersed in virtual medium outside parent virtual environment system (VES). To solve the task, the extension of the cubemap method is proposed, in which cubemap resolution is determined taking into account viewer camera field of view and screen resolution (Adequate Cubemap Projection, ACMP). The paper studies the influence of the camera orientation inside the cube on the "cubemap pixel / screen pixel" ratio determining panorama visualization quality. Based on this, a method to calculate cubemap resolution for high-quality panorama visualization for all possible camera orientations is proposed. The paper considers an efficient method and algorithm to create ACMP-video on the GPU using render-to-texture technology, which allow to synthesize panoramas with constant orientation or bound to the observer's view direction. In the research efficient methods and algorithms to play ACMP-video are also proposed, which are based on the visualization of visible cube faces and adaptive frame buffering. The obtained methods and algorithms are implemented in ACMP-video synthesis program complex (C++, OpenGL, FFmpeg) including frame capture module (embeddable into VES) and the player. The developed solution was tested in system «Virtual Earth» designed for training to observe Earth objects from the International Space Station (ISS). Using the capture module, an ACMP-video of the flight along the ISS orbit track was created. When playing this video, the trainee flies in orbit above virtual 3D surface of the Earth and can explore it by means of camera rotation. Testing of the complex confirmed the adequacy of the developed methods and algorithms to the task. The obtained scientific and practical results expand the capabilities and scope of application of VES, scientific visualization systems, video simulators and virtual laboratories; provide effective exchange of experience between researchers, etc.

Keywords: 360-degree video; omnidirectional video; cubemap projection; visualization; virtual environment.

For citation: Timokhin P.Yu., Mikhaylyuk M.V., Vozhegov E.M. Efficient methods and algorithms to synthesize 360-degree video based on cubemap projection of virtual environment. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 73–88 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–5

Acknowledgements. The publication is made within the state task on carrying out basic scientific researches (GP 14) on topic (project) «34.9. Virtual environment systems: technologies, methods and algorithms of mathematical modeling and visualization» (0065-2019-0012).

1. Введение

В настоящее время во многих важных научных и прикладных областях востребовано виртуальное окружение, в котором человек (оператор) может обучаться решению реальных задач и проводить исследования. Особенно это актуально для областей, сопряженных с работой в агрессивной и труднодоступной среде (космическая и авиационная отрасли, нефтегазовая сфера, атомная промышленность и др.) [1, 2]. Создание виртуального окружения является достаточно сложной задачей, для решения которой разрабатываются специальные программно-аппаратные комплексы – *системы виртуального окружения* (*СВО*), включающие в себя подсистемы управления, моделирования динамики, визуализации и др. [3, 4]. Часто возникает необходимость воспроизведения виртуального окружения, созданного с помощью СВО, на обычных персональных компьютерах (в том числе, мобильных и портативных устройствах), например, для анализа допущенных при обучении ошибок, обмена опытом между исследователями, демонстрации результатов перед широкой аудиторией и др. [5]. Одним из эффективных подходов является синтез панорамных видеороликов с обзором 360° (*видео 360*), при воспроизведении которых зритель может

поворачивать камеру в произвольном направлении и ощущать эффект присутствия в виртуальной среде [6]. В идеале, 360-градусная панорама является непрерывной и представляет собой сферу, в центре которой расположен наблюдатель. Чтобы получить кадр видео 360, такую «идеальную» сферу необходимо некоторым образом записать в дискретное планарное изображение (панораму 360), с которыми работают все современные видеокодеки. Ввиду того, что поверхность сферы невозможно развернуть на плоскость без искажений, обычно выбирают такие способы проецирования, при которых в областях поля зрения, наиболее важных для зрителя, искажения минимальны.

Существующие методы построения панорамы 360 можно условно разделить на две следующие группы. К *первой группе* относятся методы, в которых панорама проецируется на плоскость с помощью одной или нескольких картографических проекций. Широко распространены решения, основанные на эквидистантной цилиндрической проекции [7] (в настоящее время в данной проекции видео 360 загружается на YouTube). Эквидистантная проекция отображает панораму в истинном масштабе только вдоль экватора, а при приближении к полюсам масштаб увеличивается. Для уменьшения искажений масштаба в работе [8] было предложено проецировать сферу на плоскость горизонтальными полосами приблизительно одинакового разрешения (длины таких полос авторы сокращают при приближении к полюсам), а области полюсов записывать в виде двух круговых изображений.

Ко *второй группе* относятся методы, в которых панорама проецируется на некоторый выпуклый многогранник, который затем разворачивается на плоскость. Благодаря своей простоте и поддержке современными графическими библиотеками широкое распространение получила проекция в кубическую карту (cubemap projection, CMP) [9-12], при которой панорама отображается на 6 граней куба, где каждая грань охватывает угол обзора $90°$. В традиционной CMP все грани равнозначны (отсутствуют области полюсов с сильными искажениями масштаба), однако за счет перспективной проекции при приближении к ребрам и углам куба искажения изображения увеличиваются [10]. Для уменьшения неоднородностей детализации разрабатываются различные модификации CMP: равноугольная кубическая проекция (Equi-Angular Cubemap, EAC) [10], CMP со смещенным от центра положением наблюдателя [11], «умное» ориентирование CMP, при котором объекты переднего плана отображаются на грани, а не на ребра [12] и др. В исследовании [13] авторам удалось уменьшить потери четкости видео 360 (по сравнению с традиционной CMP) за счет проецирования панорамы на ромбододекаэдр. Разработчики из компании Facebook предложили принципиально другой подход, при котором панорама проецируется на грани пирамиды видимости наблюдателя [14]. Чтобы обеспечить охват в $360°$, авторы используют 30 таких пирамид с шагом в $30°$. Несмотря на заявленное значительное сокращение объема видеофайла (до 80% по сравнению с эквидистантной проекцией), авторы впоследствии отказались от данного подхода из-за его сложности в пользу кубической проекции.

Одной из важных задач при реализации методов, основанных на CMP, является выбор разрешения кубической карты, обеспечивающего синтез качественных изображений. При выборе слишком большого разрешения битрейт видеопотока существенно возрастает, что приводит к увеличению времени воспроизведения кадров и появлению вынужденных пауз. Если установить недостаточное разрешение, то ухудшается качество видео (возникают артефакты «блочности» изображения), в результате чего теряется эффект присутствия. В данной работе предлагаются методы и алгоритмы решения этой задачи, основанные на расширении метода кубической проекции, при котором разрешение кубической карты определяется с учетом угла обзора камеры зрителя и разрешения экрана. Такую проекцию мы называем адекватной (Adequate CMP, ACMP), а получаемое с помощью нее видео 360 – ACMP-видео. В разд. 2 предлагается метод вычисления разрешения ACMP-карты, в разд. 3 и 4 рассматриваются методы и алгоритмы создания и воспроизведения ACMP-видео, а в разд. 5 приводятся результаты апробации программного комплекса, созданного на основе предложенных методов и алгоритмов.

2. Метод вычисления разрешения ACMP-карты

Метод кубической проекции состоит в проецировании окружающей наблюдателя виртуальной среды на грани единичного куба с помощью шести перспективных камер, расположенных в центре этого куба и направленных под 90 градусов друг к другу. Каждая камера имеет вертикальный угол обзора (*FOV*), равный 90 градусов и отношение ширины к высоте формируемого изображения (*aspect*), равное единице. Таким образом, каждая камера проецирует видимое ею изображение на соответствующую грань куба. Эта проекция записывается в область вывода с некоторым разрешением d (число пикселов вдоль каждой стороны изображения).

Для просмотра результата кубического проецирования записанные изображения накладываются на грани куба, в центре куба устанавливается камера зрителя C_V, через которую он может наблюдать изображение виртуальной сцены, отображенное на гранях куба. Камера C_V может иметь произвольные *FOV* и *aspect* и ее область вывода может иметь некоторое разрешение D (число пикселов вдоль большей стороны области вывода). Для упрощения рассмотрения мысленно спроецируем области вывода на соответствующие ближние плоскости отсечения камер, так что теперь будем считать, что куб проецирования имеет размер d x d x d пикселов. Наша задача состоит в вычислении такого разрешения d кубической проекции, при котором любой отрезок пикселов из кубической проекции будет отображаться в отрезок не меньшего числа пикселов в области вывода с разрешением D.

Опишем вокруг куба сферу, имитирующую «идеальную» панораму 360, а в центр сферы поместим виртуальную камеру. Рассмотрим некоторую дугу l сферы, которая целиком попадает в поле зрения камеры и на грань куба. Построим проекцию l_{face} дуги l на грань куба (рис. 1а, в силу симметрии, на этом рисунке показана четвертая часть двумерной проекции куба). Рассмотрим также камеру C_V зрителя видео 360, и проекцию l_{cam} дуги l на ближнюю плоскость отсечения этой камеры (рис. 1б). Обозначим через α центральный угол дуги l и рассмотрим функцию $r(\eta,\beta)$ отношения длин проекций l_{face} и l_{cam} для бесконечно малой дуги l (при $\alpha \to 0$):

$$r(\eta,\beta) = \lim_{\alpha\to 0}\left(l_{face}/l_{cam}\right). \quad (1)$$

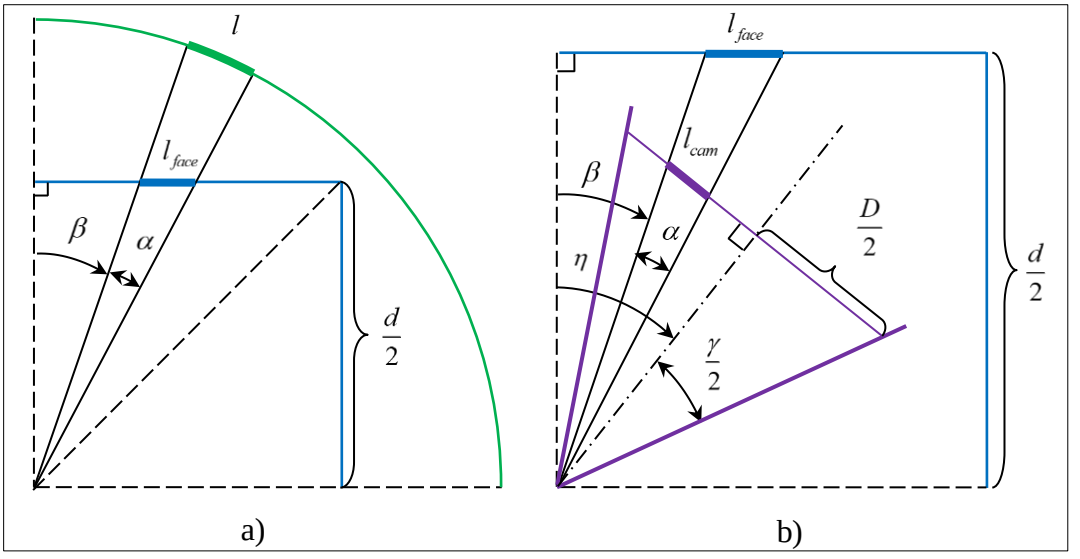

Рис. 1. Проекция дуги l: a) на грань куба; b) на ближнюю плоскость отсечения камеры
Fig. 1. Projection of arc l on: a) the face of the cube; b) near clip plane of the camera C_V

Фактически, функция r показывает, сколько пикселов изображения на грани куба (кубической проекции) будет отображаться в некоторый произвольный пиксел области вывода камеры зрителя. Во избежание образования артефактов «блочности» (когда пиксел грани куба отображается на несколько пикселов области вывода) отношение r в идеале не должно быть меньше 1 для всех пикселов экрана. На практике (например, для экранов с высокой плотностью пикселов), этот порог может быть уменьшен до некоторого значения r_{thres} без заметной потери качества видео 360. Тогда, величину d можно вычислить из равенства $r_{thres} = \min r(\eta, \beta)$.

2.1 Нахождение функции *r*

Чтобы найти функцию r, вычислим длины проекций l_{face} и l_{cam}. Введем следующие обозначения: $\beta \in [0, \pi/4 - \alpha]$ - угол отклонения дуги l от перпендикуляра к грани куба (см. рис. 1а); $\gamma \in (0, \pi)$ - угол наибольшего раствора (горизонтального или вертикального) камеры C_V (см. рис. 1б); $\eta \in [\alpha + \beta - \gamma/2, \beta + \gamma/2]$ - угол отклонения вектора взгляда камеры от перпендикуляра к грани куба.

Из рисунков 1а и 1б запишем следующие выражения для длин l_{face} и l_{cam}

$$\ell_{face} = d\frac{\mathrm{tg}(\alpha+\beta)-\mathrm{tg}\beta}{2}, \quad \ell_{cam} = D\frac{\mathrm{tg}(\eta-\beta)-\mathrm{tg}(\eta-\beta-\alpha)}{2\mathrm{tg}(\gamma/2)}. \tag{2}$$

Подставив эти выражения в формулу (1), получим

$$r(\eta,\beta) = \lim_{\alpha\to 0}\frac{l_{face}}{l_{cam}} = \frac{d}{D}\mathrm{tg}(\gamma/2)\cdot\lim_{\alpha\to 0}\frac{\mathrm{tg}(\alpha+\beta)-\mathrm{tg}\beta}{\mathrm{tg}(\eta-\beta)-\mathrm{tg}(\eta-\beta-\alpha)}. \tag{3}$$

Используя формулу преобразования разности тангенсов, приведем выражение (3) к виду

$$r(\eta,\beta) = \frac{d}{D}\mathrm{tg}(\gamma/2)\cdot\lim_{\alpha\to 0}\frac{\cos(\eta-\beta)\cos(\eta-\beta-\alpha)}{\cos(\alpha+\beta)\cos\beta}. \tag{4}$$

При уменьшении размера дуги l до бесконечно малой величины ($\alpha \to 0$) получим из (4) выражение искомой функции

$$r(\eta,\beta) = (d/D)\,\mathrm{tg}(\gamma/2)\left(\cos(\eta-\beta)/\cos\beta\right)^2, \text{ где } \beta \in [0,\pi/4],\ \eta \in [\beta-\gamma/2, \beta+\gamma/2]. \tag{5}$$

2.2 Нахождение наименьшего значения r_{min} и размера d

Легко проверить, что на указанных в (5) интервалах функция $f(\eta,\beta) = \cos(\eta-\beta)/\cos\beta$ неотрицательна, поэтому функция $r(\eta,\beta)$ принимает наименьшее значение r_{min} при наименьшем значении f_{min} функции $f(\eta,\beta)$. Чтобы найти f_{min}, попробуем сначала вычислить критические точки функции $f(\eta,\beta)$ в области $\beta \in (0,\pi/4)$, $\eta \in (\beta-\gamma/2, \beta+\gamma/2)$, т.е. те точки, в которых обе частные производные f'_η и f'_β равны нулю (это является необходимым условием наличия экстремума функции). Имеем

$$f'_\eta = -\frac{\sin(\eta-\beta)}{\cos\beta},\ f'_\beta = \frac{\sin\eta}{\cos^2\beta},\ \begin{cases} f'_\eta = 0 \\ f'_\beta = 0 \end{cases} \Rightarrow \begin{cases} \sin(\eta_{crit}-\beta_{crit}) = 0 \\ \sin\eta_{crit} = 0 \end{cases} \Rightarrow \begin{cases} \eta_{crit}-\beta_{crit} = \pi n \\ \eta_{crit} = \pi n \end{cases}. \tag{6}$$

Так как $\beta \in (0, \pi/4)$, а $\gamma \in (0, \pi)$, η не может выходить за границы интервала $(-\pi/2, 3\pi/4)$. Из этого следует, что $\eta_{crit} = 0$, а значит $\beta_{crit} = 0$. Однако это значение $\beta_{crit} \notin (0, \pi/4)$, следовательно, система (6) не имеет решения, а функция $f(\eta, \beta)$ не имеет критических точек в рассматриваемой области. Найдем теперь минимальное значение функции $f(\eta, \beta)$ на границах $\eta_1 = \beta - \gamma/2$, $\eta_2 = \beta + \gamma/2$, $\beta_1 = 0$ и $\beta_2 = \pi/4$. Получаем

$$f_1(\eta_1, \beta) = f_2(\eta_2, \beta) = \cos(\gamma/2)/\cos\beta, \quad f_3(\eta, \beta_1) = \cos\eta, \quad f_4(\eta, \beta_2) = \sqrt{2}\cos(\eta - \pi/4).$$

Функции $f_1(\eta_1, \beta)$ и $f_2(\eta_2, \beta)$ имеют наименьшее значение на отрезке $\beta \in [0, \pi/4]$, когда $\cos\beta$ максимален, т.е. при $\beta = 0$: $f_{1,\min} = f_{2,\min} = \cos(\gamma/2)$. Функция $f_3(\eta, \beta_1) = f_3(\eta, 0) = \cos\eta$ на отрезке $\eta \in [-\gamma/2, \gamma/2]$ принимает минимальное значение при $\eta = \pm\gamma/2$, откуда следует $f_{3,\min} = \cos(\gamma/2)$, а функция $f_4(\eta, \beta_2) = f_4(\eta, \pi/4) = \sqrt{2}\cos(\eta - \pi/4)$ на отрезке $\eta \in [\pi/4 - \gamma/2, \pi/4 + \gamma/2]$ имеет наименьшее значение при $\eta = \pi/4 \pm \gamma/2$, откуда $f_{4,\min} = \sqrt{2}\cos(\gamma/2)$. Так как $\gamma \in (0, \pi)$, то легко видеть, что $f_{\min} = \min_i f_i = \cos(\gamma/2)$. Подставив значение f_{min} в формулу (5), получим искомое выражение для r_{min}:

$$r_{\min} = (d/D)\,\mathrm{tg}(\gamma/2)\, f_{\min}^2 = (d/D)\,\mathrm{tg}(\gamma/2)\cos^2(\gamma/2) = (d/(2D))\sin\gamma. \quad (7)$$

Как отмечалось в начале разд. 2, размер d в данной работе находится из равенства $r_{thres} = r_{min}$. Подставим в него r_{min} из (7) и выразим из полученного равенства искомый d:

$$d = 2Dr_{thres}/\sin\gamma = 2Dr_{thres}\,\mathrm{cosec}\,\gamma. \quad (8)$$

2.3 Оценка качества визуализации кубической проекции

В формуле (8) величина r_{thres} означает наименьшее возможное число пикселов грани куба, которое будет отображаться в любой пиксел камеры зрителя, т.е. фактически r_{thres} определяет уровень качества визуализации кубической проекции в камере зрителя. Мы будем считать значение $r_{thres} = 1$, соответствующим наибольшему уровню качества, т.к. дальнейшее повышение числа отображаемых пикселов грани куба не будет иметь смысла - мы их просто не сможем различить в камере зрителя. Наибольший уровень качества необходим для отображения кубической проекции в камере зрителя с невысоким разрешением D, и, наоборот, для высоких значений D (Full HD, 4K) уровень качества может быть уменьшен. Рассмотрим это на следующем примере.

Вычислим по формуле (8) величину d для $\gamma = 50°$ (угол, близкий к восприятию пространства человеческим зрением), $D = 1024$ пикселов и $r_{thres} = 1$. При этих параметрах мы получим значение $d \approx 2600$ пикселов. Подставим значения d, D и γ в формулу (5) и построим поверхность $r(\eta, \beta)$ (см. рис. 2). Полученная поверхность-«седло» показывает изменение уровня качества визуализации кубической проекции в зависимости от места расположения объекта наблюдения (дуги l сферы 360 градусов) на грани куба (угол β) и в камере зрителя (угол η). На рисунке 2 передний торец «седла» соответствует случаю, когда объект наблюдения располагается в центре грани куба ($\beta = 0$), а задний торец – когда объект наблюдения находится по направлению в ребро куба ($\beta = \pi/4$). При этом самые верхние точки торцевых изгибов соответствуют случаю, когда камера зрителя направлена в объект наблюдения ($\eta = 0$ и $\eta = \pi/4$), т.е. объект находится в центре камеры, а самые нижние

точки - расположению объекта у границ раствора камеры зрителя (для переднего торца - $\eta = \pm\gamma/2 = \pm 25°$, а для заднего - $\eta = \pi/4 - \gamma/2 = 20°$ и $\eta = \pi/4 + \gamma/2 = 70°$).

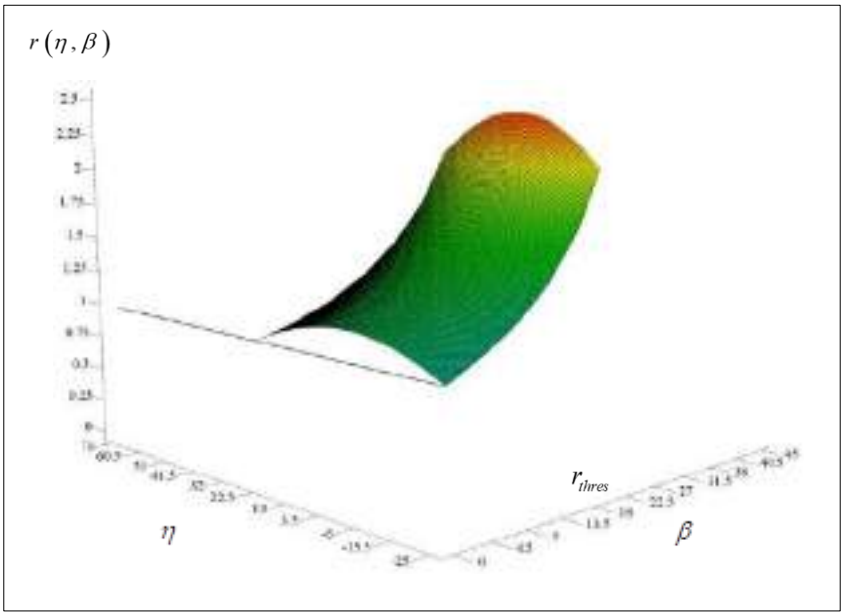

Рис. 2. Пример поверхности $r(\eta, \beta)$ при визуализации кубической проекции с высоким качеством
Fig. 2. Example of $r(\eta, \beta)$ surface at high-quality visualization of cubemap projection

Из рисунка 2 видно, что даже в самом худшем случае (объект в центре грани куба и у границы раствора камеры, см. концы переднего торца «седла») кубическая проекция будет визуализироваться с высоким качеством: в любой пиксел камеры зрителя будет отображаться не менее 1 пиксела грани куба. На практике, построение кубической проекции с разрешением граней d = 2600 пикселов может потребовать больших вычислительных ресурсов, например, если виртуальная сцена содержит сложные высокополигональные текстурированные объекты. В таких случаях можно пойти на уменьшение уровня r_{thres} качества до значения, позволяющего решать задачу, для которой создается видео 360 (на рис. 2 при уменьшении r_{thres} «седло» будет параллельно сдвигаться вниз). Так, например, при r_{thres} = 0.5, объекты в центральной части поля зрения камеры будут визуализироваться с качеством не менее 75%, а в периферийной – с качеством не менее 50% (при этом разрешение граней d уже составит около 1300 пикселов).

3. Метод создания ACMP-видео

Рассмотрим задачу создания видео 360 для наблюдателя, который исследует виртуальное окружение с помощью жестко связанной с ним виртуальной камеры C_{obs}. Наблюдатель может перемещаться и поворачивать камеру. Мы будем синтезировать кадр видео 360 на основе кубической проекции с разрешением d, вычисленным согласно (8), каждые $1/f_v$ миллисекунд, где $f_v \geq 25$ - частота смены кадров видеоролика в кадрах в секунду (d и f_v задаются с помощью диалогового окна до начала записи видео 360). Рассмотрим процесс синтеза кадра для некоторого такого момента времени.

Обозначим через P_{obs}, $\boldsymbol{v}_{obs}$ и $\boldsymbol{up}_{obs}$ позицию, вектор взгляда и вектор «вверх» камеры C_{obs} в мировой системе координат (World Coordinate System, WCS) в этот момент времени. Поместим в точку P_{obs} шесть одинаковых виртуальных камер $C_0, \ldots, C_5$ с FOV = 90°, aspect = 1 и областью вывода d x d пикселов. Направление камеры C_1 совпадает с направлением камеры C_{obs}, а остальные камеры расположены под 90 градусов друг к другу (см. рисунок 3).

Направим эти камеры так, чтобы их ближние плоскости образовывали куб, как показано на рисунке 3 (грани куба названы так, как их видит наблюдатель изнутри).

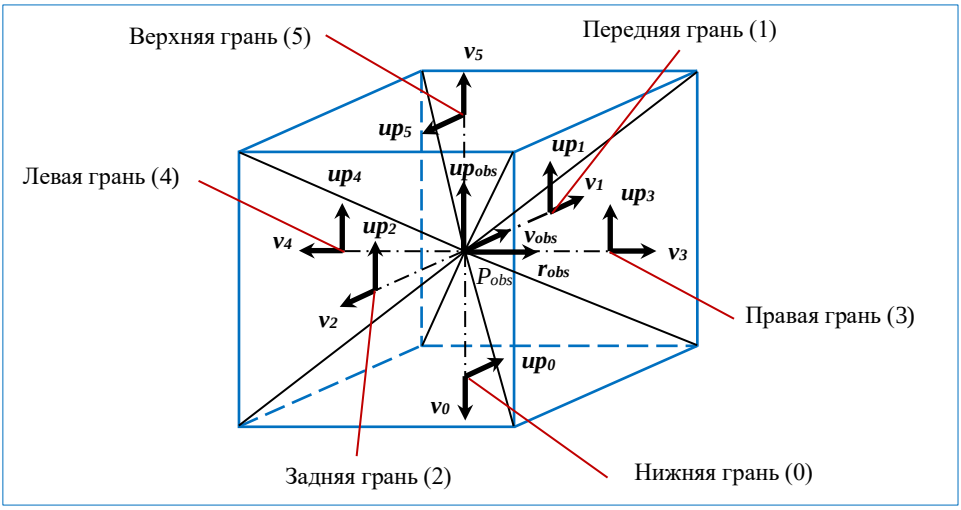

Рис. 3. Построение кубической проекции
Fig. 3. Constructing cubemap projection

Векторы $\boldsymbol{v}_0,\ldots,\boldsymbol{v}_5$ взгляда и векторы $\boldsymbol{up}_0,\ldots,\boldsymbol{up}_5$ «вверх» камер $C_0,\ldots,C_5$ всегда коллинеарны одному из векторов $\boldsymbol{v}_{obs}$, $\boldsymbol{up}_{obs}$, $\boldsymbol{r}_{obs}$, где $\boldsymbol{r}_{obs} = \boldsymbol{v}_{obs} \times \boldsymbol{up}_{obs}$ (вектор «вправо» камеры C_{obs}), а их координаты находятся в соответствии с таблицей 1.

Таблица 1. Координаты векторов камер C_i
Table 1. Coordinates of camera C_i vectors

Грань	***0 (нижняя)***	***1 (передняя)***	***2 (задняя)***	***3 (правая)***	***4 (левая)***	***5 (верхняя)***
$\boldsymbol{v}_i$	$-\boldsymbol{up}_{obs}$	$\boldsymbol{v}_{obs}$	$-\boldsymbol{v}_{obs}$	$\boldsymbol{r}_{obs}$	$-\boldsymbol{r}_{obs}$	$\boldsymbol{up}_{obs}$
$\boldsymbol{up}_i$	$\boldsymbol{v}_{obs}$	$\boldsymbol{up}_{obs}$	$\boldsymbol{up}_{obs}$	$\boldsymbol{up}_{obs}$	$\boldsymbol{up}_{obs}$	$-\boldsymbol{v}_{obs}$

Каждый синтезируемый кадр ACMP-видео будет состоять из 6 изображений, полученных из камер C_i. Синтез такого кадра будем выполнять в RGB-текстуре T размера $3d$ x $2d$ пикселов. Поставим в соответствие каждой C_i-ой камере участок d x d пикселов (тайл) текстуры T, как показано на рис. 4.

1. Активируем FBO и очистим текстуры T и U с помощью операторов `glClearColor, glClearDepth` и `glClear`.
2. Зададим матрицу перспективной проекции (общую для камер $C_0,\ldots,C_5$) с FOV = 90° и aspect = 1 с помощью оператора `gluPerspective`.
3. Цикл по i от 0 до 5
   Установим область вывода C_i-ой камеры размера d x d пикселов с координатами левого нижнего угла $x = (i \,\%\, 3)d$, $y = \lfloor i/3 \rfloor d$ с помощью оператора `glViewport`.
   Зададим матрицу видового преобразования C_i-ой камеры с позицией P_{obs} и векторами $\boldsymbol{v}_i$ и $\boldsymbol{up}_i$ с помощью `gluLookAt`.
   Выполним рендеринг виртуальной сцены из камеры C_i.
   Конец цикла.
4. Деактивируем FBO.

Алгоритм 1. Синтез ACMP-кадра
Algorithm 1. ACMP-frame synthesis

С помощью графической библиотеки OpenGL создадим внеэкранный буфер кадра (Framebuffer Object, FBO) [15] с текстурами T (буфер цвета) и U (буфер глубины). Далее

выполним рендеринг (с включенным тестом глубины) виртуальной сцены из каждой C_i-ой камеры в i-ый тайл текстуры T с помощью *Алгоритма 1*.

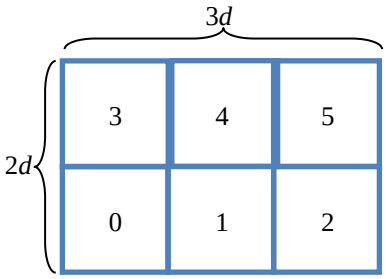

Рис. 4. Тайлы АСМР-кадра
Fig. 4. Tiles of ACMP-frame

После выполнения *Алгоритма 1* полученный кадр АСМР-видео (заполненная текстура T) выгружается из видеопамяти в оперативную память, сжимается и добавляется в видеопоток с помощью набора библиотек FFmpeg [16]. Работа со сжатыми видеопотоками более подробно описана в [5].

4. Метод воспроизведения АСМР-видео

Для воспроизведения АСМР-видео в данной работе создается виртуальная 3D сцена (см. рис. 5), содержащая модель единичного куба с центром в WCS, в который помещается виртуальная камера C_V зрителя (введена в разделе 2). Повороты камеры C_V разрешены только вокруг осей X и Y свой локальной системы координат, что соответствует наклону головы вверх/вниз и влево/вправо. Воспроизведение АСМР-видео включает в себя следующие ключевые шаги: (а) считывание и декодирование АСМР-кадров с частотой f_v; (б) извлечение из очередного АСМР-кадра тайлов-текстур для граней куба, видимых из камеры C_V; (в) плавная визуализация АСМР-кадров. Описание шага (а) можно найти в работе [5], поэтому далее рассмотрим шаги (б) и (в).

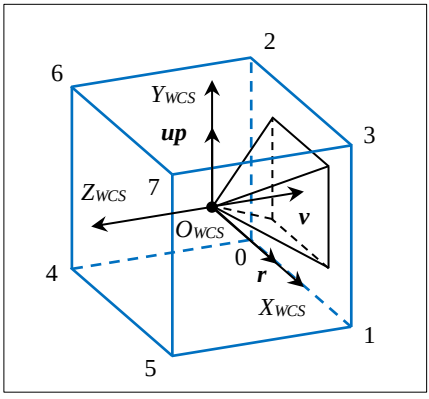

Рис. 5. Единичный куб и камера C_V
Fig. 5. Unit cube and camera C_V

4.1 Отбор текстур видимых граней куба

Для определения тайлов-текстур (граней куба), необходимых для визуализации АСМР-кадра, введем понятие *двухгранника* – пары граней куба с общим ребром. Как можно заметить, у куба 12 вариантов двухгранников (по числу ребер). Пронумеруем вершины куба, как показано на рисунке 5 и запишем массив D ребер куба: {0, 1}, {1, 5}, {5, 4}, {4, 0}, {6, 4}, {7,

5}, {3, 1}, {2, 0}, {2, 3}, {3, 7}, {7, 6}, {6, 2}. Для ребер из массива D запишем массив E вариантов двухгранников, используя нумерацию граней куба из рисунка 3: {0, 1}, {0, 3}, {0, 2}, {4, 0}, {2, 4}, {2, 3}, {3, 1}, {1, 4}, {1, 5}, {3, 5}, {5, 2}, {4, 5}. В зависимости от параметров камеры C_V при просмотре видео в область вывода могут попадать ребра куба, а могут и не попадать. Ребра, попадающие в поле зрения камеры C_V, будут определять двухгранники, которые необходимо использовать при визуализации. Если же в поле зрения не попадает ни одного ребра, то камера C_V будет захватывать какую-то одну грань куба.

Чтобы упростить проверку видимости ребер куба, создадим таблицу H булевских флагов положения каждой вершины куба относительно каждой из 5 отсекающих плоскости камеры (0 – ближняя, 1 – левая, 2 – правая, 3 – нижняя, 4 – верхняя). Дальнюю плоскость отсечения мы не учитываем, т.к. выставляем ее заведомо дальше всех вершин куба. Таблица H состоит из 8 строк, где номер строки совпадает с номером вершины куба. Каждая строка хранит флаги $b_0,\ldots,b_4$, а также флаг $b_5 = (b_0 \,\&\&\, b_1 \,\&\&\, b_2 \,\&\&\, b_3 \,\&\&\, b_4)$. Истинный b_i-ый флаг ($i = 0,\ldots,4$) означает, что вершина куба лежит в i-ой отсекающей плоскости или ее «+» полупространстве, а истинный флаг b_5 - что вершина куба находится в пирамиде видимости камеры C_V. Вычисление флагов описано в работе [17].

С помощью таблицы H будем отбирать для визуализации каждый двухгранник, общее ребро которого пересекает пирамиду видимости камеры C_V. Обозначим через b_{pair} флаг наличия хотя бы одного отобранного двухгранника (*true*/*false* - есть/нет). Грани отобранных двухгранников (или видимую одиночную грань куба) будем отмечать в булевском массиве B_{faces} длины 6 (по числу граней в кубе, *true*/*false* - грань видна/не видна). Это реализует *Алгоритм 2*.

1. Очистим массив B_{faces} значением *false*, b_{pair} = *false*.
2. Цикл по j от 0 до 11, где j – индекс ребра куба
   Если ($H_{D[j][0],\,5}$ || $H_{D[j][1],\,5}$), то b_{pair} = *true*; // видима хотя бы одна вершина ребра.
   В противном случае (вершины ребра не видимы), проверим, пересекает ли ребро хотя бы одну отсекающую плоскость, и видима ли точка P_u их пересечения:
   Цикл по i от 1 до 4, где i - отсекающая плоскость камеры C_V
      Если ($H_{D[j][0],\,i}$ ^ $H_{D[j][1],\,i}$), то: // ^ – исключающее ИЛИ.
         Вычислим координаты точки P_u и ее флаг b_5 видимости (см. [17]).
         Если b_5 равен *true*, то: $B_{faces}[E[j][0]]$ = *true*, $B_{faces}[E[j][1]]$ = *true*, b_{pair} = *true*, выходим из цикла.
   Конец цикла.
3. Если в камере C_V не видно ни одного ребра куба (b_{pair} = *false*), то ищем k-ую грань, у которой угол между внешней нормалью и вектором $\boldsymbol{v}$ камеры C_V будет наименьшим:
   k = 0, запишем массив K косинусов углов между внешними нормалями к граням куба и вектором $\boldsymbol{v}$ взгляда: $K = \{-\boldsymbol{v}_y, -\boldsymbol{v}_z, \boldsymbol{v}_z, \boldsymbol{v}_x, -\boldsymbol{v}_x, \boldsymbol{v}_y\}$.
   Цикл по p от 1 до 5, где p - номер грани куба.
      Если $K[i] > K[k]$, то $k = i$.
   Конец цикла.
   $B_{faces}[k]$ = *true*.

Алгоритм 2. Получение номеров видимых граней куба
Algorithm 2. Getting the numbers of visible cube faces

В результате выполнения *Алгоритма 2* в элементах массива B_{faces}, соответствующих видимым граням куба, будут записаны значения *true*. По номеру n каждого такого элемента мы находим смещения $offsetX = (n\%3)d$ и $offsetY = \lfloor n/3 \rfloor d$ (в пикселах) тайла-текстуры n-ой видимой грани относительно левого нижнего угла ACMP-кадра (см. рис. 6) и загружаем эту

тайл-текстуру в видеопамять (в массив из 6 текстурных объектов, созданный перед визуализацией). Чтобы уменьшить время простоя GPU, мы загружаем каждую тайл-текстуру в видеопамять одним непрерывным куском (а не построчно) с помощью связки операторов *glPixelStorei* [15]. Отметим, что если ACMP-кадр не меняется, например, когда видео на паузе, то мы не выполняем повторную загрузку тайлов в текстурные объекты.

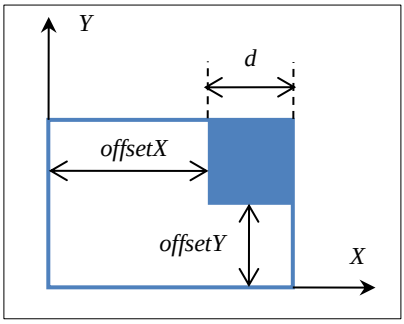

Рис. 6. Чтение тайла ACMP-кадра
Fig. 6. Reading a tile of ACMP-frame

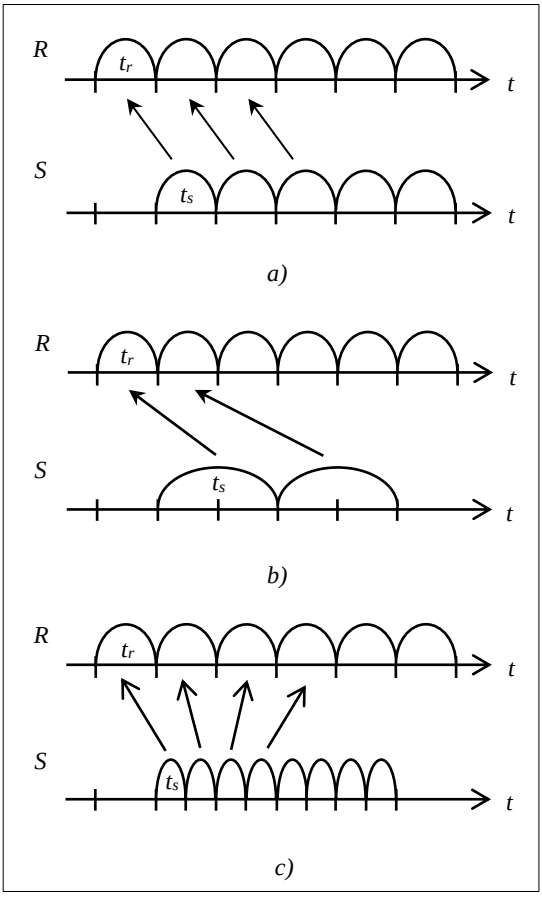

Рис. 7. Совместная работа рендер-потока и потока подкачки при: a) $t_r = t_s$; b) $t_r < t_s$; c) $t_r > t_s$
Fig. 7. Co-working of render-thread and swap-thread at: a) $t_r = t_s$; b) $t_r < t_s$; c) $t_r > t_s$

4.2 Визуализация ACMP-кадров

Чтобы визуализировать ACMP-кадр, необходимо загрузить его сжатый образ из видеофайла в оперативную память (RAM) и декодировать в RGB-изображение. Это занимает определенное время, которое препятствует плавности визуализации ACMP-кадров. Чтобы решить эту задачу в данной работе реализуется подкачка (загрузка в RAM и декодирование) впередистоящих ACMP-кадров в *кольцевой буфер* B_{ring} длины L, рассчитываемой с учетом размеров ACMP-видео, а также производительности используемых CPU и GPU.

Подкачка ACMP-кадров в буфер B_{ring} выполняется в параллельном потоке S (swap-thread), а управление потоком S осуществляет основной поток R визуализации (render-thread) на основе соотношения времен t_s и t_r подкачки и визуализации текущего ACMP-кадра. При этом обрабатываются следующие 3 возможных варианта совместной работы этих потоков. В *первом* варианте ($t_r = t_s$) пока используется текущий элемент (ACMP-кадр) буфера B_{ring}, в его предыдущий элемент подкачивается новый ACMP-кадр (см. рис. 7a). Данный случай является идеальным и не требует корректировки совместной работы потоков. Во *втором* варианте ($t_r < t_s$) рендер-поток выполняется быстрее и в какой-то момент догоняет поток подкачки (см. рис. 7b). В этом случае реализуется приостановка рендер-потока до полного заполнения буфера B_{ring} потоком подкачки (иначе начнут повторно воспроизводиться уже проигранные ACMP-кадры). *В третьем* варианте ($t_r > t_s$) возникает обратная ситуация: поток подкачки выполняется быстрее и в некоторый момент времени догоняет рендер-поток (см. рис. 7c). В этой ситуации реализуется приостановка потока подкачки до появления в буфере B_{ring} свободных для записи элементов (проигранных ACMP-кадров), иначе будут перезаписаны еще не проигранные ACMP-кадры.

5. Результаты

На основе предложенных методов и алгоритмов был разработан программный комплекс синтеза ACMP-видео виртуального окружения. Комплекс включает в себя *модуль захвата* ACMP-кадров, встраиваемый в подсистему визуализации СВО, а также *плеер* созданных ACMP-видео. Данное решение реализовано на языке C++ с применением графической библиотеки OpenGL и инструментария FFmpeg по кодированию/декодированию видео.

a) *b)*

Рис. 8. Апробация программного комплекса синтеза ACMP-видео: a) кадр из ACMP-видео полета вдоль трассы МКС; b) воспроизведенная из кадра панорама Земли

Fig. 8. Approbation of the program complex for ACMP-video synthesis: a) a frame from ACMP-video of the flight along ISS orbit; b) Earth panorama reconstructed from the frame

Созданное решение было апробировано в системе «Виртуальная Земля» [18] тренировки наблюдения объектов земной поверхности с орбиты Международной космической станции (МКС). С помощью модуля захвата было создано ACMP-видео полета вдоль трассы 2-го

суточного витка МКС для средства наблюдения с углом обзора ~ 40° и областью вывода 10^3x10^3 пикселов.

На рисунке 8a показан пример кадра, полученного ACMP-видео. Синтез ACMP-видео выполнялся с разрешением 3000x2000 пикселов и частотой 25 кадров/с на основе стандарта H.264 сжатия видео и контейнера MP4 (оба из международного стандарта MPEG-4). На рисунке 8b показано воспроизведение полученного ACMP-видео с помощью разработанного плеера. В процессе воспроизведения обучаемый совершает полет по орбите МКС, во время которого может исследовать виртуальную трехмерную поверхность Земли, поворачивая камеру в произвольном направлении. Апробация разработанного программного комплекса проводилась на компьютере (Intel Core i5 2.5 ГГц, 8 Гб RAM, 250Гб SSD), оборудованном видеокартой GeForce GTX 1050Ti (4Гб VRAM, 768 ядер).

Апробация показала, что при разрешении до 10^3x10^3 пикселов на грань куба создание ACMP-видео может выполняться в онлайн режиме, т.е. непосредственно в процессе работы СВО. Чтобы создавать ACMP-видео с более высоким разрешением, в будущем планируется развить предложенные методы и алгоритмы в направлении отложенной записи видеороликов [5], при которой в онлайн режиме захватываются только динамические параметры виртуальной сцены (сценарий), а преобразование сценария в видеоролик выполняется уже в офлайн режиме.

6. Заключение

В статье рассмотрена задача создания и воспроизведения качественного панорамного видео с обзором 360 градусов на основе проекции виртуального окружения в кубическую карту. Эвристический выбор разрешения карты является проблематичным, т.к. при недостаточном разрешении ухудшается качество видео и теряется эффект погружения в виртуальную среду, а при избыточном - возрастает битрейт видеопотока и возникают задержки воспроизведения. Для решения этой проблемы в статье предложен ACMP-подход (Adequate Cubemap Projection), при котором разрешение кубической карты определяется с учетом угла обзора камеры зрителя и разрешения экрана. В работе выведена функция зависимости отношения «пиксел карты/пиксел экрана» от ориентации камеры, характеризующая качество визуализации панорамы. На основе исследования этой функции был разработан метод вычисления разрешения кубической карты для качественной визуализации панорамы при всех возможных ориентациях камеры.

В работе предложены эффективные метод и алгоритм создания ACMP-видео на GPU, основанные на технологии рендеринга в текстуру, которые позволяют выполнять построение кубических панорам с привязкой к направлению взгляда наблюдателя. Также в данной статье предложены эффективные методы и алгоритмы воспроизведения ACMP-видео, основанные на отборе для визуализации видимых участков ACMP-кадра и адаптивной буферизации ACMP-кадров в параллельном потоке. Полученные методы и алгоритмы были реализованы в программном комплексе, который включает в себя модуль захвата ACMP-кадров и плеер ACMP-видео.

Разработанный комплекс был апробирован в системе «Виртуальная Земля» тренировки наблюдения объектов земной поверхности с орбиты МКС. Было создано ACMP-видео, с помощью которого обучаемый может вне родительской СВО выполнить полет по орбите МКС над виртуальной трехмерной поверхностью Земли и исследовать ее, поворачивая камеру. Апробация комплекса подтвердила адекватность разработанных методов и алгоритмов поставленной задаче и выявила пути их дальнейшего развития. Полученные научные и практические результаты могут быть применены в системах виртуального окружения, имитационно-тренажерных комплексах, системах научной визуализации и виртуальных лабораториях, образовательных приложениях и др.

Список литературы / References

[1]. Михайлюк М.В., Мальцев А.В., Тимохин П.Ю., Страшнов Е.В., Крючков Б.И., Усов В.М. Системы виртуального окружения для прототипирования на моделирующих стендах использования космических роботов в пилотируемых полетах. Пилотируемые полеты в космос, том 35, № 2, 2020 г., стр. 61-75. / Mikhaylyuk M.V., Maltsev A.V., Timokhin P.Yu., Strashnov E.V., Kryuchkov B.I., Usov V.M. Virtual Environment Systems for Simulating Robots in Manned Space Fligts. Pilotiruemye polety v kosmos. Manned Spaceflight, vol. 35, № 2, 2020, pp. 61-75 (in Russian).

[2]. Барладян Б.Х., Шапиро Л.З., Маллачиев К.А., Хорошилов А.В., Солоделов Ю.А., Волобой А.Г., Галактионов В.А., Ковернинский И.В. Система визуализации для авиационной ОС реального времени JetOS. Труды ИСП РАН, том 32, вып. 1, 2020 г., стр. 57-70. DOI: 10.15514/ISPRAS-2020-32(1)-3 / Barladian B.Kh., Shapiro L.Z., Mallachiev K.A., Khoroshilov A.V., Solodelov Y.A., Voloboy A.G., Galaktionov V.A., Koverninskiy I.V. Rendering System for the Aircraft Real-Time OS JetOS. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 1, 2020, pp. 57-70 (in Russian).

[3]. Михайлюк М.В., Торгашев М.А. Система визуализации “GLView” для имитационно-тренажерных комплексов и систем виртуального окружения. Труды 25-й Международной научной конференции GraphiCon, 2015, стр. 96-101 / Mikhaylyuk M.V., Torgashev M.A. The System of Visualization “GLView” for Simulators and Virtual Environment Systems. In Proc. of the 25th International Conference on Computer Graphics and Vision (GraphiCon 2015), 2015, pp. 96-101 (in Russian).

[4]. Страшнов Е.В., Мироненко И.Н., Финагин Л.А. Моделирование режимов полета квадрокоптера в системах виртуального окружения. Информационные технологии и вычислительные системы, № 1, 2020 г., стр. 85-94 / Strashnov E.V., Mironenko I.N., Finagin L.A. Simulation of quadcopter flight modes in virtual environment systems. Informacionnye tekhnologii i vichslitel’nye sistemy (Journal of Information Technologies and Computing Systems), № 1, 2020. pp. 85-94 (in Russian).

[5]. Тимохин П.Ю., Михайлюк М.В., Вожегов Е.М., Пантелей К.Д. Технология и методы отложенного синтеза 4K-стереороликов для сложных динамических виртуальных сцен. Труды ИСП РАН, том 31, вып. 4, 2019 г., стр. 61-72. DOI: 10.15514/ISPRAS-2019-31(4)-4. / Timokhin P.Yu., Mikhaylyuk M.V., Vozhegov E.M., Panteley K.D. Technology and methods for deferred synthesis of 4K stereo clips for complex dynamic virtual scenes. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 4, 2019, pp. 61-72 (in Russian).

[6]. El-Ganainy T., Hefeeda M. Streaming Virtual Reality Content. arXiv:1612.08350, 2016..

[7]. Ray B., Jung J., Larabi M.-C. A Low-Complexity Video Encoder for Equirectangular Projected 360 Video Content. In Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2018, pp. 1723-1727

[8]. Li J., Wen Z., Li S., Zhao Y., Guo B., Wen J. Novel tile segmentation scheme for omnidirectional video. In Proc. of the IEEE International Conference on Image Processing (ICIP), 2016, pp. 370-374.

[9]. K.-T. Ng, S.-C. Chan, H.-Y. Shum. Data Compression and Transmission Aspects of Panoramic Videos. IEEE Transactions on Circuits and Systems for Video Technology, vol. 15, № 1, 2005, pp. 82-95

[10]. Brown C. Bringing pixels front and center in VR video. Google AR and VR, March 14, 2017. Available at: https://www.blog.google/products/google-ar-vr/bringing-pixels-front-and-center-vr-video/, accessed 18.03.2020.

[11]. Kuzyakov E., Liu S., Pio D. Optimizing 360 Video for Oculus. Facebook F8 developers conference, 2016. Available at: https://developers.facebook.com/videos/f8-2016/optimizing-360-video-for-oculus/, accessed 18.03.2020.

[12]. Chen Z., Wang X., Zhou Y., Zou L., Jiang J. Content-Aware Cubemap Projection for Panoramic Image via Deep Q-Learning. Lecture Notes in Computer Science, vol. 11962, 2020, pp. 304-315.

[13]. Fu C.-W., Wan L., Wong T.-T., Leung C.-S. The Rhombic Dodecahedron Map: An Efficient Scheme for Encoding Panoramic Video. IEEE Transactions on Multimedia, vol. 11, № 4, 2009, pp. 634-644.

[14]. Kuzyakov E., Pio D. Next-generation video encoding techniques for 360 video and VR. Available at: https://code.facebook.com/posts/1126354007399553/next-generation-video-encodin, accessed 18.03.2020.

[15]. Segal M., Akeley K. The OpenGL Graphics System: A Specification. Version 4.6, Core Profile. The Khronos Group Inc., 2006-2018. Available at: https://www.khronos.org/registry/OpenGL/specs/gl/glspec46.core.pdf, accessed 18.03.2020.

[16]. FFmpeg. A complete, cross-platform solution to record, convert and stream audio and video. Available at: https://ffmpeg.org/, accessed 18.03.2020.

[17]. Timokhin P.Y., Mikhaylyuk M.V. Effective technology to visualize virtual environment using 360-degree video based on cubemap projection. In Proc. of International Conference on Computing for Physics and Technology (CPT2020), 2020.

[18]. Тимохин П.Ю. Моделирование видимого движения Земли вдоль участков суточной трассы МКС в космических видеотренажерах. Труды НИИСИ РАН, том. 9, № 6, 2019 г., стр. 111-117. DOI: 10.25682/NIISI.2019.6.0014 / Timokhin P.Yu. Simulation of visible Earth motion along daily tracks of ISS orbit in space simulators. Trudy NIISI RAN/Proc. of SRISA RAS, vol. 9, № 6, 2019, pp. 111-117 (in Russian).

Информация об авторах / Information about authors

Петр Юрьевич ТИМОХИН – старший научный сотрудник. Сфера научных интересов: компьютерная графика, визуализация.

Petr Yurievich TIMOKHIN – Senior Researcher. Research interests: computer graphics, visualization.

Михаил Васильевич МИХАЙЛЮК – доктор физико-математических наук, профессор, главный научный сотрудник. Сфера научных интересов: компьютерная графика, визуализация, системы виртуального окружения.

Mikhail Vasilievich MIKHAYLYUK – Doctor of Physical and Mathematical Sciences, Professor, Chief Researcher. Research interests: computer graphics, visualization, virtual environment systems.

Евгений Михайлович ВОЖЕГОВ – ведущий программист. Сфера научных интересов: компьютерная графика.

Evgeniy Mikhaylovich VOZHEGOV – Leading programmer. Research interests: computer graphics.