УДК 004.92

Ядров М.В.
магистрант 1 курса
факультет автоматики и вычислительной техники
кафедра автоматизированных систем управления
Новосибирский государственный технический университет
(г. Новосибирск, Россия)

Научный руководитель:
Ландовский В.В.
кандидат технических наук, доцент
кафедра автоматизированных систем управления
Новосибирский государственный технический университет
(г. Новосибирск, Россия)

КОНСОЛИДАЦИЯ ТЕКСТУРНЫХ РЕСУРСОВ КАК ИНСТРУМЕНТ ОПТИМИЗАЦИИ 3D-СЦЕН

Аннотация: *в статье предлагается авторский алгоритм консолидации текстур на основе анализа и выявления общих закономерностей в наборах графических данных, а также приводится обоснование эффективности путём теоретических расчётов затрачиваемой памяти.*

Ключевые слова: *консолидация текстур, текстурный ресурс, оптимизация 3D-сцены, алгоритм обработки, анализ графических данных, оценка потребления памяти.*

Анализ предметной области.

Идея сжатия исходных графический данных для уменьшения занимаемого ими места не нова – один из первых алгоритмов появился в 1979 и опирался на разбиение изображений по блокам с последующим усреднением значений яркости в каждой из половин блока [1]. Со временем подход к

оптимизации изображений сильно изменился: появились методы векторного квантования (1996) и аппаратно поддерживаемые форматы блочного сжатия (S3TC, 1997, PVRTC, 2003, ETC, 2004, ASTC, 2012). Параллельно развивались методы организации и сокращения объёма текстурных данных, включая атласирование, виртуализацию и процедурный синтез текстур. Современное направление исследований связано с нейросетевым сжатием (NTC, 2023), позволяющим совместно кодировать несколько карт материала с использованием компактных обучаемых представлений [2].

Однако, методы оптимизации не ограничиваются лишь сжатием текстур. В низкополигональных сценах допускается использование текстур-палитр, поскольку все объекты на сцене имеют примитивный вид и простую раскраску, то допустимо весь набор цветов представить небольшой текстурой, к которой будет обращаться каждый объект сцены – рисунок 1.



Рисунок 1. Пример использования палитры в low-poly сцене.

Так как все объекты сцены ссылаются на один и тот же исходник, эти объекты можно объединить в один, тогда вся сцена будет представлена одной командой отрисовки (draw-call). Подобный подход пока что не может быть применён к сложным сценам из-за объёма и разнообразия визуальных данных.

Простая «склейка» сложных текстур допустима (уже упомянутые атласы), но она не является инструментом уменьшения объёма данных – только альтернативным форматом хранения. В большинстве случаев выигрыш от использования атласов нивелируется сложностью хранения визуальных данных и сложностью отсечения невидимой геометрии.

Обзор существующих инструментов и алгоритмов.

Алгоритмы не могут существовать сами по себе – в современных средствах разработки и обработки компьютерной графики представлен широкий спектр инструментов оптимизации текстурных данных. Основное внимание в них уделяется алгоритмам сжатия текстур, направленным на сокращение объёма занимаемой памяти при сохранении приемлемого визуального качества (таблицы 1 и 2). Вместе с тем существуют инструменты консолидации текстур, позволяющие объединять несколько текстурных изображений в единые атласы для более эффективного использования графических ресурсов (таблица 3). Подобные средства реализованы не только в игровых движках, таких как Unity и Unreal Engine, но и в специализированных программных решениях для обработки текстур и трёхмерной графики.

Таблица 1. Инструменты сжатия текстур движков.

Среда	Инструменты	Возможности
Unity	Texture Import Settings, Texture Compression, Crunch Compression	BCn, ASTC, ETC, платформенные настройки качества и сжатие файлов
Unreal Engine 5	Texture Editor, Texture Compression Settings, Oodle Texture	BCn, ASTC и другие платформенные форматы, оптимизация кодирования
Godot 4	Import Dock, VRAM Compressed, Basis Universal	GPU-сжатие, универсальное сжатие с транскодированием
CryEngine	Resource Compiler, Texture Compiler	Подготовка текстур и платформенно-зависимое сжатие

Таблица 2. Специальные инструменты текстурного сжатия.

Инструмент	Поддерживаемые технологии	Назначение
AMD Compressonator	BCn, ASTC и другие форматы	Сжатие, сравнение качества, анализ ошибок
NVIDIA Texture Tools Exporter	BCn, DDS, KTX	Экспорт GPU-текстур, обработка изображений
ARM ASTC Encoder (astcenc)	ASTC LDR/HDR	Настраиваемое ASTC-сжатие, выбор размера блока и качества
PVRTexTool	PVRTC, ETC, ASTC и др.	Подготовка текстур для мобильных GPU
Basis Universal	ETC1S, UASTC	Универсальное сжатие с последующим транскодированием
Khronos KTX-Software	KTX2, Basis Universal, GPU-форматы	Создание, кодирование и проверка переносимых текстур

Таблица 3. Инструменты консолидации текстур.

Среда	Инструмент	Возможности
Unity	Sprite Atlas	Объединение 2D-спрайты в атласы
Unity	Texture2D.PackTextures()	Программная упаковка несколько текстур в одну
Unreal Engine 5	Runtime Virtual Texturing	Кэширование результатов затенения и объединение данных материалов в виртуальном текстурном пространстве
Unreal Engine 5	Streaming Virtual Texturing	Динамическая загрузка необходимых страниц больших текстур
Blender	UV Editing, Pack Islands, Cycles Bake	Позволяет подготовить общий UV-атлас и запечь в него текстуры нескольких объектов
Adobe Substance 3D Painter	Texture Set Export, Channel Packing	Экспортирует наборы текстур и объединяет разные карты материала по каналам

Анализ рассмотренных инструментов позволяет выделить ряд общих особенностей применяемых подходов к оптимизации текстурных данных. Большинство средств сжатия ориентировано на использование аппаратно-поддерживаемых форматов, обеспечивающих компактное хранение текстур и возможность их эффективного декодирования графическим процессором. При этом выбор алгоритма сжатия определяется требованиями к качеству изображения, объёму занимаемой памяти и возможностями целевой платформы. Средства консолидации текстур, в свою очередь, преимущественно направлены на объединение нескольких изображений в общие атласы, что позволяет сократить количество используемых текстурных ресурсов и потенциально уменьшить число переключений материалов при визуализации. Следует отметить, что сжатие и консолидация, несмотря на возможность их совместного применения, в большинстве рассмотренных решений реализованы как самостоятельные операции, требующие отдельной настройки и обработки данных.

Для количественной оценки эффективности инструментов можно сравнить объём памяти, необходимый для хранения текстурных изображений при использовании различных форматов сжатия. В качестве примера возьмём текстуру разрешением 2048×2048 пикселей с четырьмя цветовыми компонентами по 8 бит каждый (RGBA8). В несжатом виде каждый пиксель занимает 32 бита, что позволяет определить исходный объём текстурных данных по формуле:

$$V_{исх} = \frac{W * H * B}{8}$$

где W и H – ширина и высота изображения в пикселях, B – количество бит на пиксель. Для рассматриваемой текстуры исходный объём составляет:

$$V_{исх} = \frac{2048 * 2048 * 32}{8} = 16777216B = 16MB.$$

Средства сжатия текстур, представленные в Unity, Unreal Engine и специализированных программах, таких как AMD Compressonator и ARM ASTC Encoder, позволяют преобразовывать исходные изображения в аппаратно поддерживаемые блочные форматы. В форматах BC1, BC3 и BC7 используются блоки размером 4×4 пикселя, занимающие соответственно 64, 128 и 128 бит. Формат ASTC предусматривает выбор размеров блока при фиксированном объёме 128 бит на блок, благодаря чему достигается различная плотность хранения данных.

Объём текстуры после сжатия:

$$V_{сж} = \frac{W * H * B_b}{w_b * h_b * 8}$$

Результаты расчёта для наиболее распространённых вариантов сжатия/способов хранения приведены в таблице 4.

Таблица 4. Результаты сжатия текстуры.

Формат	**Бит/пиксель**	**Объём, МиБ**
RGBA8	32	16,00
BC1	4	2,00
BC3	8	4,00
BC7	8	4,00
ASTC 6×6	≈3,56	≈1,78
ASTC 8×8	2	1,00

Для определения эффективности алгоритмов используются коэффициент сжатия (K) и относительное сокращение объёма памяти (S):

$$K = \frac{V_{исх}}{V_{сж}} \; S = \left(1 - \frac{V_{сж}}{V_{исх}}\right) * 100\%.$$

Таблица 5.

Формат	**Коэффициент**	**Экономия памяти**
BC1	8:1	87,50%
BC3	4:1	75,00%
BC7	4:1	75,00%
ASTC 6×6	≈8,96:1	≈88,84%
ASTC 8×8	16:1	93,75%

Оценка потери качества изображений.

При сжатии с потерями исходные значения пикселей заменяются приближёнными, что может приводить к размытию деталей, изменению цветовых переходов и появлению блочных артефактов (рисунок 2). Для количественной оценки подобных искажений используются среднеквадратичная ошибка (MSE) и пиковое отношение сигнал/шум (PSNR).

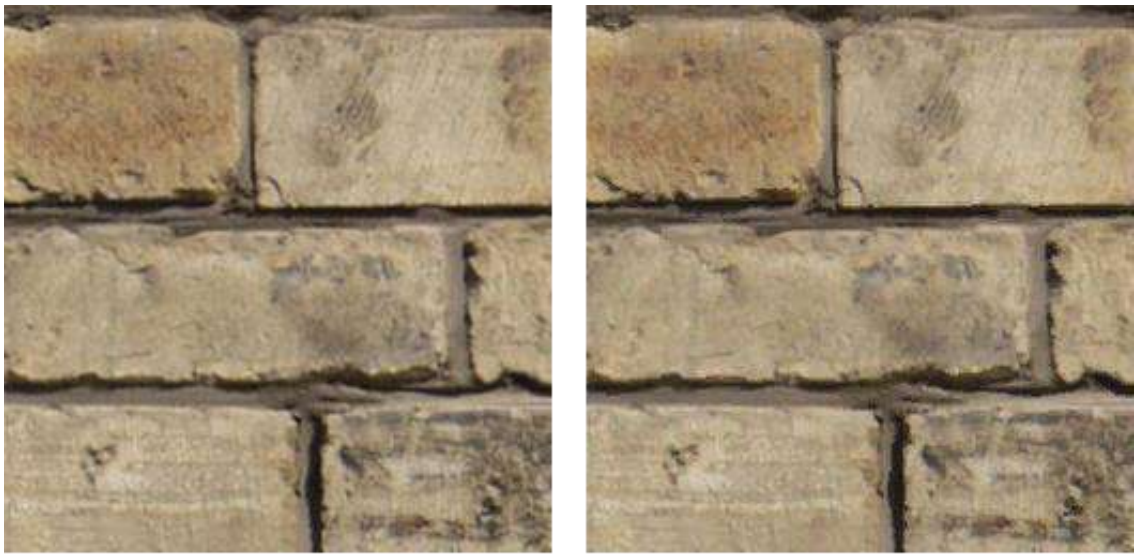


Рисунок 2. Пример искажения текстуры после восстановления.

Среднеквадратичная ошибка определяется выражением:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2$$

где N – количество сравниваемых значений, x_i – исходное значение компонента изображения, $\hat{x}_i$ – восстановленное значение.

Пиковое отношение сигнал/шум вычисляется по формуле:

$$PSNR = 10log_{10}\left(\frac{MAX^2}{MSE}\right)$$

где MAX=255 для восьмибитного представления.

Чем меньше значение MSE и выше PSNR, тем точнее восстановленное изображение соответствует исходному по среднеквадратичной ошибке. При этом PSNR не полностью отражает субъективное восприятие качества, поэтому итоговая оценка остаётся за пользователем или зрителем. Для наглядности был проведён перевод нескольких текстур в форматы BC1, BC3 и BC7 с расчётом MSE и SPNR, результаты на рисунке 3.

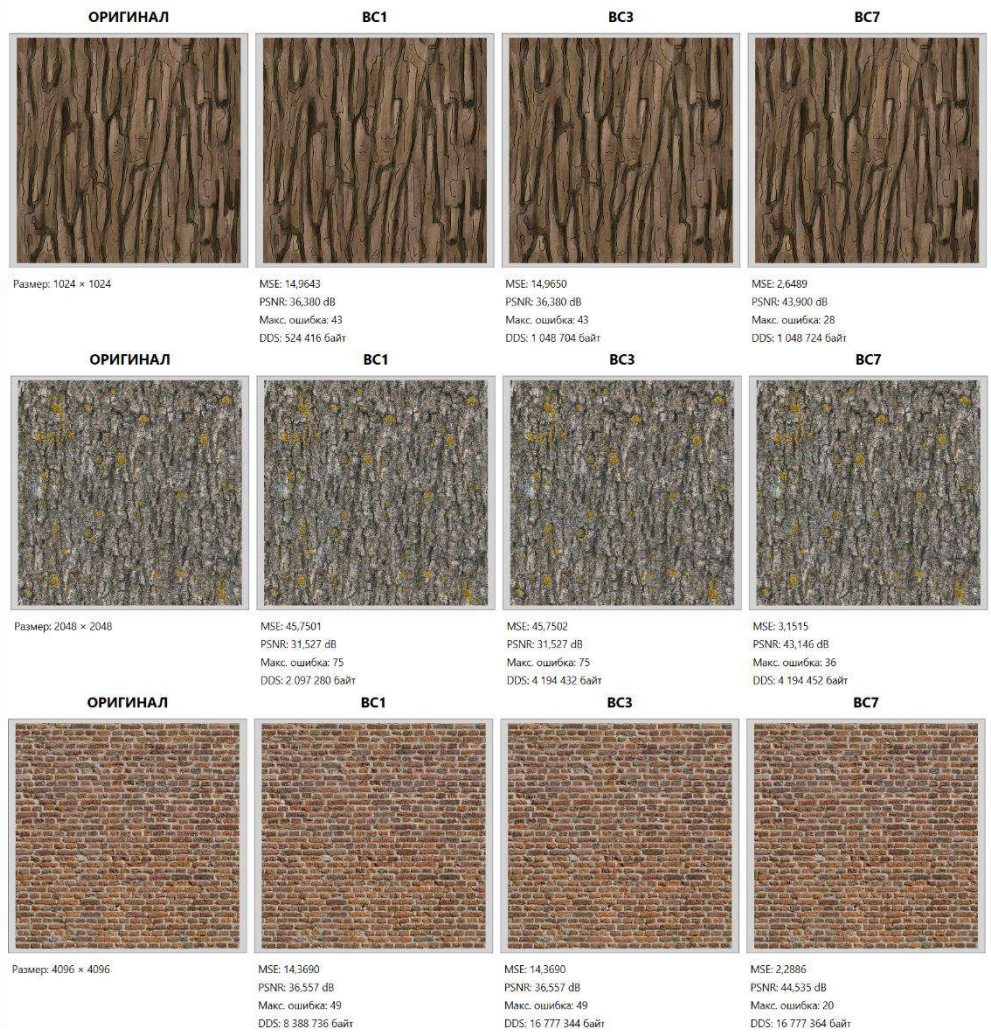


Рисунок 3. Сравнение изменения качества текстур при конвертации в разные поколения Block Compression.

Можно заметить, что явного изменения внешнего вида, как и ухудшения качества изображения, не наблюдается. Происходит это из-за особенностей человеческого восприятия изображений. Глаз видит текстуру не как набор

пикселей, а как комбинацию перцептивных признаков: грани объектов, цветовые переходы, распределения яркости и т.д [3].

Поскольку текстуры ориентированы в первую очередь на человеческую перцепцию, логично было бы разработать алгоритм, опирающийся именно на распознавание ключевых визуальных признаков изображений с их последующей консолидацией для снижения занимаемого места графической составляющей 3D сцен.

Разработка алгоритма.

Начать стоит с перевода текстуры в блочное представление. Поэтапный анализ блоков пикселей с постепенным уменьшением размеров рассматриваемых фрагментов позволит создать иерархический атлас классификации элементов текстуры – рисунок 4.

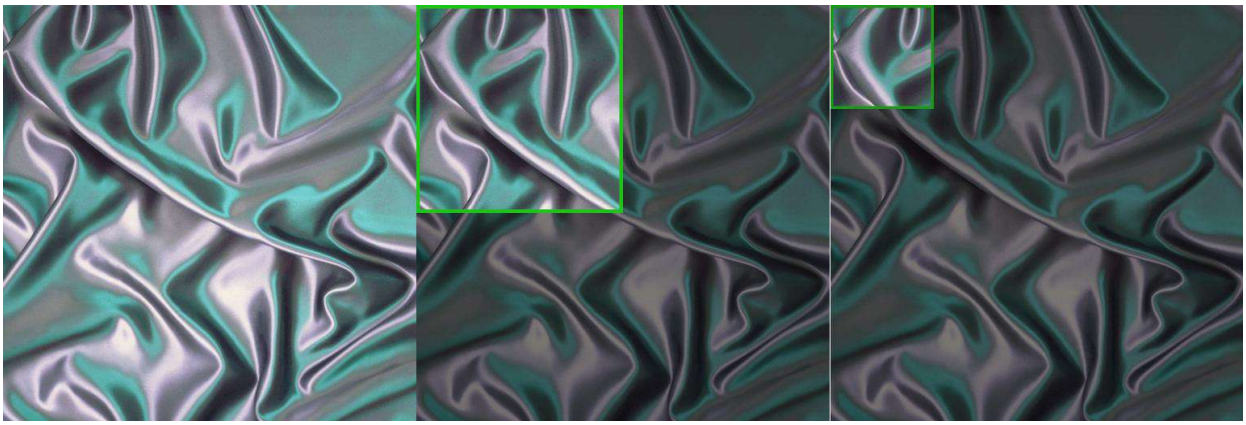


Рисунок 4. Пример разбиения изображения на фрагменты.

Прекратить уменьшение фрагментов можно на любом этапе, пока алгоритм не упрётся в минимальную допустимую конструкцию – квадрат 2*2 пикселя. Для текстуры 1024*1024 максимальный размер атласа будет составлять 9 слоёв и 34952 квадратов всех размеров.

Для более удобного сравнения текстур разных цветов стоит использовать не сами численные значения окраски пикселя, а их среднее. Каждый квадрат на каждом уровне проходит визуальную классификацию по внутренним паттернам,

список которых приведён в таблице 6. Саму классификацию допустимо провести через векторное сканирование фрагмента [4], пример на рисунке 5.

Таблица 6. Перечень графических паттернов.

Группа	Паттерны
Градиенты	Однородная заливка, Горизонтальный, Вертикальный, Диагональный, Радиальный центральный, Радиальный смещённый, Двусторонний, Нелинейный монотонный
Границы	Прямая граница, Размытая граница, Смещённая граница, Диагональная граница, Т-образное соединение, Крест
Полосы	Параллельные полосы, Неравномерные полосы, Плавные синусоидальные полосы, Волнистые линии, Кольцо/кольца, Линейные перекрытия, Прерывающиеся линии, Хаотические линии
Регулярные геометрические структуры	Прямоугольная решётка, Шахматное замощение, Точечная решётка, Гексагональная решётка, Нарушенная решётка
Зернистые и дискретные структуры	Равномерная зернистость, Случайная зернистость, Изолированные пятна, Сгруппированные пятна, Чешуйчатая структура, Фрагментированная структура
Шумовые и случайные структуры	Белый шум, Гауссовский шум, Импульсный шум, Коррелированный шум, Низкочастотный шум, Высокочастотный шум, Анизотропный шум
Иерархические и многомасштабные структуры	Масштабно-самоподобная структура, Иерархическая вложенность, Многомасштабная зернистость, Изменяющаяся пространственная частота, Изменяющийся размер элементов, Изменяющаяся плотность элементов, Иерархическое ветвление

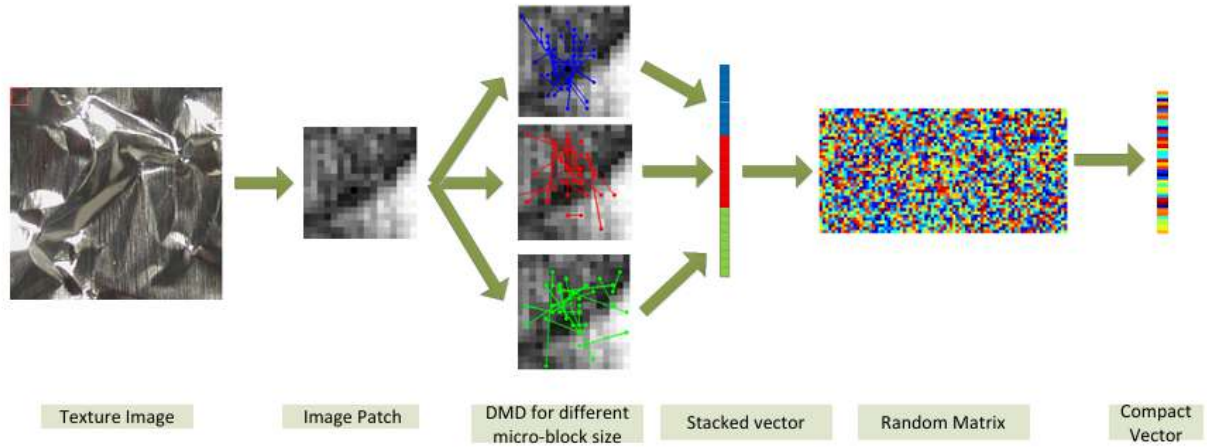


Рисунок 5. Пример векторного сканирования (взято из [4]).

После того, как все текстуры сцены будут приведены к блочному формату, можно начать формирование функций цветового восстановления. Как это сделать: каждый блок хранит в себе средние значения пикселей, через которые можно восстановить приблизительные исходные значения простыми операциями сложения и вычитания, пример на рисунке 6 – слева исходные цвета, их конкретные значения RGB и средние, в скобках – номера функций восстановления, справа – восстановленные оттенки.



Рисунок 6. Пример восстановления цветов через средние значения.

Предел максимального размера функции восстановления ограничен масштабом фрагмента, в котором ведётся цветовой анализ: для 2*2 предел равен 3, для 4*4 – 5 и по нарастающей. Стоит отметить, что после завершения

консолидации останутся только самые эффективные функции восстановления – те, что дают наименьшую погрешность, остальные же будут удалены для экономии памяти.

Следующий этап – поиск кандидатов для консолидации. Обычный перебор всех возможных вариантов комбинаций блоков и текстур слишком не эффективен, поэтому нужно ввести несколько ограничений. Первое – консолидация должна проходить по всей стороне текстуры, это нужно для корректной работы последующих преобразований в мипмапы и перевода в другие форматы изображений – рисунок 7.

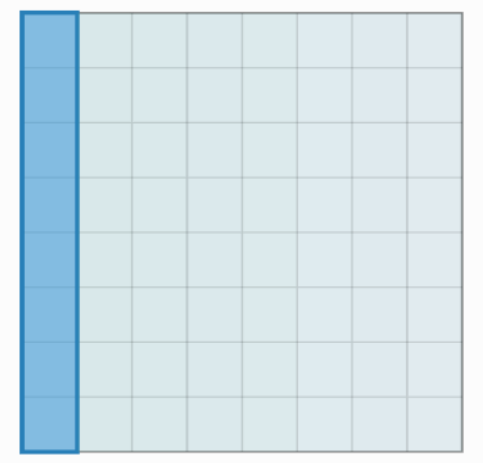


Рисунок 7. Пример минимальной зоны консолидации.

Из первого ограничения следует второе – расширение зоны консолидации может быть только по всей стороне вглубь текстуры. По итогу будет 4 основных направления роста: слева-направо, справа-налево, сверху-вниз и снизу-вверх. Третье ограничение регулирует темп роста консолидируемой зоны – следующий шаг может быть не больше предыдущего или достаточно большим для увеличения зоны на площадь, равную нескольким предыдущим расширениям. Например, для текстуры 1024*1024 стартовый размер зоны консолидации равнялся 128*1024, следующий может так же равен 128*1024, либо может быть меньше: 64*1024, 32*1024 и т.д., в случае повторного выбора 128*1024 максимальная площадь третьего расширения может быть выбрана 256*1024,

потому что уже консолидированная зона представляет собой фрагмент (128+128)*1024. Четвёртое ограничение отсекает потенциально бесполезные зоны консолидации – если всё же удалось найти общие визуальные признаки у двух текстур, но по итогу выделенная область оказалась незначительно малой, то нет смысла выделять её в отдельную сущность, создавать функции восстановления и переназначать вызовы рендера объектов (подробнее о качестве консолидации в главе «Расчёты эффективности»).

После завершения обработки текстур, консолидированные зоны отсекаются и становятся новыми объектами, хранящими: атлас средних значений пикселей, оставшиеся функции восстановления, связи с полигонами моделей на сцене, которые будут ссылаться на эту визуальную информацию. Все данные о ненужных функциях восстановления, классификации текстурных блоков и маршрутах расширения зон консолидации также удаляются за ненадобностью.

Расчёт эффективности.

Для определения условий эффективного применения алгоритма необходимо рассчитать общий объём памяти, занимаемый графической составляющей сцены после консолидации:

$$S_{new} \approx \frac{4}{3} * q * P + S_m + S_r$$

где q – доля значений, оставшихся после консолидации, P – изначальное количество текселей всех текстур, S_m – таблицы областей, ссылок и функций восстановления, S_r – объём дополнительных данных для коррекции ошибок, коэффициент 4/3 – объём уменьшенных версий исходных изображений (применяются при отдалении объекта от наблюдателя).

Относительно BC1 и BC7 разумно применять алгоритм, если:

$$q \leq a - \frac{S_m + S_r}{4P/3}$$

где a – коэффициент разности объёма памяти для представления цвета – 1 для BC7 и 0,5 для BC1.

Можно выделить 3 типовых вида сцен: примитивные, состоящие из простых объектов с набором таких же простых текстур (рисунок 8), средние, хранящие в себе сложные геометрические образы с соответствующим цветовым оформлением (рисунок 9) и максимально детализированные (рисунок 10).



Рисунок 8. Пример примитивной сцены.



Рисунок 9. Пример средней сцены.



Рисунок 10. Пример максимально детализированной сцены.

В первом случае за исходные данные будет взят набор из 256 текстур по 512*512 пикселей, примеры на рисунке 11.



Рисунок 11. Примеры текстур примитивной сцены.

Можно заметить, что разнообразия визуальных признаков нет, некоторые текстуры просто копируют одну карту яркости для разных цветов (в примере это текстуры грязи и травы), что отлично подходит для консолидации. Изначальный объём текстур будет равен примерно 85 МБ, после оптимизации с коэффициентом q=0.3 – уже 30.6 МБ, если брать объём функций восстановления и ссылок на уровне 5 МБ, что даёт экономию в 64.1% относительно BC7 и только 28.3% относительно BC1.

Во втором случае текстуры имеют выраженное визуальное различие, поэтому коэффициент q увеличится. Верхней допустимой границей для q можно считать 0.7, поскольку при этом значении имеется видимое уменьшение объёма текстур и сохраняется оптимальный объём выделяемой под функции восстановления и ссылочные таблицы памяти. 256 текстур по 1024*1024 пикселей займут 256 мегабайт, после упрощения с максимальным допустимым q результат будет составлять примерно 205 мегабайт с учётом выделения 25 из них на вспомогательные данные.

Для максимально детализированной сцены берётся набор из 1024 текстур по 2048*2048 пикселей, итоговый объём равен 4096 МБ, из-за сложности консолидации q=0.9, причём необходимый объём для вспомогательных данных увеличиться из-за большого количества пикселей в консолидированных зонах: 3686.4+300+100=4086, что почти равно изначальному объёму, поэтому применение консолидации в этом случае не имеет особого значения.

Заключение.

Предварительные результаты оказались неоднозначными. В текущей формулировке алгоритм отлично справляется с упрощением и без того простых и нетребовательных сцен, однако, при работе со сложной графикой использование консолидации может приводить к малозначимым изменениям, если не к ухудшению результата.

Нерешённым остаётся вопрос допустимости консолидации при разности паттернов некоторых блоков линии, также необходимо провести дополнительную оценку затрачиваемого на дополнительные данные объёма памяти, помимо этого нужно ввести внутренние параметры определяемых графических закономерностей для более качественного поиска фрагментов для консолидации.

Уже сейчас, с учётом всех недостатков, алгоритм имеет потенциал в области оптимизации 3D сцен.

СПИСОК ЛИТЕРАТУРЫ:

1. Delp E., Mitchell O. Image compression using block truncation coding // IEEE Transactions on Communications. 1979. V. 27. № 9. P. 1335–1342 https://doi.org/10.1109/tcom.1979.1094560
2. Nystad J. et al. Adaptive scalable texture compression // Proceedings of the Fourth ACM SIGGRAPH/Eurographics Conference on High-Performance Graphics. 2012. P. 105–114
3. Kurosawa F. et al. Textures vs non-textures: A simple computational method for classifying perceived «texturality» in natural images // i-Perception. 2021. V. 12. № 5. P. 1–7 https://doi.org/10.1177/20416695211054540
4. Mehta R., Egiazarian K. Texture classification using dense micro-block difference // IEEE Transactions on Image Processing. 2016. V. 25. № 4. P. 1604–1616 https://doi.org/10.1109/tip.2016.2526898

Yadrov M.V.
master's degree student of 1st year
Faculty of Automation and Computer Engineering
Department of Automated Control Systems
Novosibirsk State Technical University
(Novosibirsk, Russia)

Scientific advisor:
Landovsky V.V.
PhD in Engineering, Associate Professor
Department of Automated Control Systems
Novosibirsk State Technical University
(Novosibirsk, Russia)

TEXTURE RESOURCE CONSOLIDATION AS A TOOL FOR 3D SCENE OPTIMIZATION

Abstract: *the article proposes an original texture consolidation algorithm based on identifying common patterns in graphical data sets. Its effectiveness is supported by theoretical calculations of memory consumption.*

Keywords: *texture consolidation, texture resource, 3D scene optimization, processing algorithm, graphical data analysis, memory consumption assessment.*