Выбор генератора псевдослучайных чисел для использования в рендеринге методом трассировки пути

И.Ю. Сесин, В.В. Нечаев

***Аннотация* — В статье рассматривается набор практических требований, предъявляемых к генераторам псевдослучайных чисел, используемых в программных реализациях метода рендеринга, известного как метод трассировки пути. Список сформированных требований направлен на достижение наиболее качественной работы программной реализации рендерера и избежание типовых проблем, возникающих в ходе внедрения рендереров подобного типа.**

***Ключевые слова* — генератор псевдослучайных чисел, рендеринг, трассировка пути.**

I. Введение

Рендерингом называют процесс получения растрового изображения исходя из некоторой информации о трехмерной сцене. Рендеринг является одним из столпов компьютерной графики и широко применяется в кинематографе и компьютерных играх.

В связи с возрастающими вычислительными мощностями компьютеров, доступных рядовому пользователю, в области компьютерной графики наблюдается рост интереса к наиболее ресурсоёмким техникам рендеринга, ранее представлявшим исключительно академический интерес.

В частности, одним из подобных методов рендеринга является метод трассировки пути, позволяющий получать качественные фотореалистичные изображения, но, при этом характеризующийся исключительной ресурсоёмкостью в отношении вычислительных мощностей ЭВМ.

В процессе реализации рендерера возникает множество задач, прямо не следующих из теоретического описания алгоритма, и, зачастую, достаточно комплексных и многогранных, чтобы требовать многократной итерации решений, прежде чем будет найден оптимум.

Одной из таких задач, возникающих в курсе построения программной реализации трассировки пути, и совершенно неочевидных с точки зрения чисто теоретического подхода к рендерерам, является задача подбора алгоритма генерации псевдослучайных чисел.

Статья получена 29 июля 2017 г.

И.Ю. Сесин, МИРЭА (e-mail: ghostwithashotgun@gmail.com).

В.В. Нечаев, к.т.н. профессор, зав. лаб. МИРЭА (e-mail: nechaev@mirea.ru).

II. Уравнение рендеринга и трассировка пути

В 1986 году Джеймс Кадзийя представил уравнение рендеринга в своей работе [1] как модель поведения света в трехмерной сцене:

$$L_O(x,\omega,\lambda,t) = L_e(x,\omega,\lambda,t) + \int_\Omega f_r(x,\omega',\omega,\lambda,t)L_i(x,\omega',\lambda,t)(-\omega' \cdot n)d\omega', \quad (1)$$

где:

λ – длина волны света,

t – время,

$L_O(x,\omega,\lambda,t)$ – количество излучения с длиной волны λ, выпущенного из точки x вдоль направления ω в момент времени t,

$\int_\Omega f_r(x,\omega',\omega,\lambda,t)L_i(x,\omega',\lambda,t)(-\omega' \cdot n)d\omega'$ – интеграл по полусфере входящего излучения,

$f_r(x,\omega',\omega,\lambda,t)$ – двулучевая функция отражательной способности (ДФОС), описывающая процесс, подобно тому, как свет отражается от поверхности,

$L_i(x,\omega',\lambda,t)$ – сумма входящего излучения, зависящая от длины волны λ излучения, выпущенного в точку x по направлению ω' в момент времени t,

$-\omega' \cdot n$ – скалярное произведение направления $-\omega'$ и нормали к поверхности n, характеризующее поглощение излучения при отражении.

В общем случае, это интегральное уравнение постулирует, что количество испущенного из точки x излучения L_O равно сумме излученного света L_e и отраженного света (сумма всего входящего излучения по полусфере L_i на коэффициент поглощения).

Очевидно, решать данное интегральное уравнение аналитически для любого невырожденного случая – нерационально, поэтому на практике применяют численные методы. В частности, Джеймс Кадзийя при дальнейшем рассмотрении полезности уравнения в своей работе, помимо других методов, описал применение метода Монте-Карло для вычисления интеграла входящего излучения по полусфере. Именно этот подход лег в основу метода трассировки пути.

Трассировка пути считается методом рендеринга без допущений, в связи с использованием метода Монте-Карло, то есть при увеличении количества итераций он будет сходиться к ответу.

III. Генераторы псевдослучайных чисел

Так как для вычисления интеграла входящего

излучения по полусфере используется метод Монте-Карло, возникает задача получения большого количества (не менее 1×10^6) случайных чисел.

Стоит отметить, что типовой пользовательский компьютер является детерминированным и не может произвести получение действительно случайных чисел без использования либо внешних источников (ресурсы интернет, или иные каналы связи), либо специального аппаратного обеспечения, использующего природные и физические процессы, характеризующиеся высокой случайностью (дробовой шум, тепловой шум, метеорологические явления).

Вместо этого используются псевдослучайные числа, генерируемые специализированными программными генераторами псевдослучайных чисел (ГПСЧ).

Существуют различные принципы организации ГПСЧ, но одним из самых основных является подход, в котором ГПСЧ имеет особую переменную, содержащую биты энтропии и называемую внутренним состоянием. Это состояние в ходе работы генератора претерпевает изменения (англ. permutation). При таком подходе после каждого изменения внутреннее состояние используется для получения результирующего псевдослучайного числа. При этом ГПСЧ требует инициализации этого внутреннего состояния при помощи некоторого исходного значения – так называемого зерна (англ. seed).

Сам процесс получения исходного внутреннего состояния из зерна зависит от частной реализации генератора. Иногда переданным генератору зерном заполняют биты внутреннего состояния, иногда применяется хеширование зерна для получения внутреннего состояния с предопределенным количеством битов, и гораздо менее предсказуемой энтропией. Очень часто примеры программного кода реализации ГПСЧ оставляют вопрос инициализации внутреннего состояния открытым.

Следует отметить, что механизм функционирования ГПСЧ является детерминированным, то есть, один и тот же генератор, инициализированный одним и тем же зерном, будет производить одинаковую последовательность псевдослучайных чисел. В рамках задачи рендеринга методом трассировки пути это свойство прямым образом не используется, однако о нём следует помнить, так как этот эффект может спонтанно проявиться при определенных подходах к инициализации ГПСЧ.

IV. Ограничения, накладываемые на ГПСЧ

Для целей рендеринга с использованием стохастических моделей, как правило, советуют выбрать «качественный ГПСЧ», не утруждаясь объяснением, что есть параметр «качества» применимо к ГПСЧ. Параметр качества может быть сформулирован различными способами, начиная от экспертной оценки и заканчивая сравнением с статистическими эталонами. Поэтому вместо абстрактного параметра качества предлагается руководствоваться рядом требований, специфичных для области рендеринга и продиктованных опытом на практике.

Вопрос правильного выбора ГПСЧ для рендерера является неожиданно сложным, в ретроспективе можно даже сказать, коварным, в связи с тем, что ни одно из выдвинутых в данной статье требований не следует прямым образом из теории и вовлеченного математического аппарата, и это выясняется только в процессе реализации алгоритма на практике.

Первый набор требований к ГПСЧ можно определить, рассмотрев типовую практическую реализацию рендерера методом трассировки пути. В связи с исключительной ресурсоёмкостью данного метода, как правило, применяется распараллеливание алгоритма и его вынесение для обсчета на графический процессор (видеокарту).

Очевидным образом, все части задействованного алгоритма должны выполняться с высокой скоростью, чтобы программа в целом могла выполниться за приемлемое время. Соответственно, первое, самое очевидное ограничение, накладываемое на ГПСЧ – ГПСЧ должен работать быстро. Скорость ГПСЧ достаточно трудно определить точно. При наиболее грубом и общем сравнении очень часто работает правило – чем меньше операций, тем быстрее работа генератора. Для более точного определения скорости требуется проведение специализированного тестирования производительности ГПСЧ.

В типовой реализации, на одну итерацию трассировки участка пути будет приходиться, в среднем, от трех до пяти вызовов ГПСЧ.

Учитывая, что числа, сгенерированные ГПСЧ, прямым образом оказывают влияние на поведение луча (в случае вычисления угла рассеивания) и ветвление программы (стохастическая модель поверхности), данные вычисления не могут быть отложены или корректно распараллелены, исключая возможность их оптимизации такими методами.

При этом, в силу устройства архитектуры графического процессора, любое обращение к оперативной памяти будет занимать большое количество времени. Из этого следует, что в данном случае нельзя применить оптимизационный метод, берущий на вооружение тактику однократной предварительной генерации огромного массива случайных чисел и запись их в память для последующего многократного использования уже в процессе рендеринга, который мог бы эффективно работать в условиях рендеринга на центральном процессоре.

В силу данных обстоятельств, использование типовых криптографических генераторов категорически не рекомендуется, несмотря на их заманчивые характеристики в отношении периода повторения, распределения, и прочих.

Вместо этого следует обращать особое внимание непосредственно на скорость работы алгоритма генератора, стремясь найти наиболее оптимальное соотношение скорости и качества. Как будет уточнено далее, для данных целей хоть и требуются высококачественные случайные числа, но планка требуемого качества далеко не настолько высока, как в

задачах криптографии.

Следует так же помнить, что на графическом процессоре, в отличие от центрального процессора, есть ряд тонкостей при работе с оперативной памятью. В частности, сама модель памяти при работе с видеокартой разделена на несколько уровней, каждый из которых характеризуется скоростью доступа и количеством доступной памяти. При этом, чем больше памяти доступно на конкретном уровне, тем дольше к нему доступ. Наиболее быстро можно обращаться к регистрам, а доступ к оперативной памяти процесса – наиболее медленный. Из этого следует второе требование, предъявляемое к ГПСЧ – внутреннее состояние должно быть достаточно малым, чтобы тысячи параллельно запущенных потоков могли работать без ограничений. Оптимальным с точки зрения скорости является вариант, когда внутреннее состояние генератора настолько мало, что может быть сохранено в регистрах.

Именно поэтому использование криптографических генераторов псевдослучайных чисел будет вносить дополнительные временные издержки в работу программы, так как для обеспечения криптографической стойкости криптографические ГПСЧ требуют немалое внутреннее состояние, которое, к тому же постоянно требуется перезаписывать. При этом нельзя произвести оптимизацию посредством использования общего внутреннего состояния для всех исполняемых потоков, так как операция взятия числа и изменения внутреннего состояния не является атомарной, и без введения механизма семафоров и блокирования доступа к памяти на время итерации ГПСЧ будут происходить непредсказуемые ошибки в работе генератора. Далее, если учесть, что параллельно будет пытаться осуществить доступ к внутреннему состоянию около нескольких тысяч потоков, то это становится узким местом всей программы, приводя к колоссальным потерям времени из-за фактически «схлопывания» параллельного алгоритма в последовательный. В результате вывод – для оптимальной скорости следует использовать ГПСЧ с малым внутренним состоянием, которое один исполняемый поток может унести «на себе», т.е. в регистровой памяти вместо его хранения в оперативной памяти.

Помимо этого, из формул, используемых для получения случайных равномерно распределенных направлений в пределах полусферы [2], прямо следует, что псевдослучайные числа, генерируемые ГПСЧ, должны подчиняться равномерному закону распределения. Соответственно, следует использовать ГПСЧ, который выдаёт равномерно распределённые числа.

Так же существует следующая тонкость: ряд ГПСЧ могут выдавать статистически равномерный результат, но при этом их использование приводит к ухудшению качества картинки. Это связано с тем, что на действительно больших выборках (десятки миллионов сгенерированных чисел) генератор действительно даёт равномерный результат, однако на меньших выборках, при выполнении некоторого ряда условий, генератор может выдавать числа, подчиняющиеся иному закону распределения, сводя на нет свойство методов рендеринга без допущений сходиться к решению. В качестве грубого, но наглядного примера, можно привести следующую ситуацию. Если генератор, производящий числа от 1 до 10 в серии из миллиона выборок в качестве первых ста тысяч чисел возвратил единицу, а в качестве вторых ста тысяч – двойку и так далее, то строго говоря, у него равномерное распределение. Однако, продуктивно пользоваться таким генератором совершенно невозможно. Иногда свойство генератора возвращать подчиняющиеся распределению числа на малоразмерных выборках называют непредсказуемостью, тем не менее, это определение не всегда корректно, особенно если речь заходит о криптографических ГПСЧ. Итак, следует тщательно проверять, что ГПСЧ ведет себя как следует на выборках, размер которых соответствует среднему количеству выбранных случайных чисел на один путь.

Схожим по эффекту вопросом является подбор таких ГПСЧ, которые быстро выходят из вырожденных состояний. Дело в том, что при неудачном зерне ГПСЧ попадает в так называемое «вырожденное состояние», и при вызове он будет возвращать однообразные числа в течение целого ряда итераций. Иначе говоря, определенные ГПСЧ при плохом внутреннем состоянии возвращают предсказуемые числа. Обычно это не играет особой роли в областях применения ГПСЧ, однако в задаче трассировки пути этот момент является критическим. Это связано с тем, что в типичной реализации, первые три числа (для угла азимута, угла высоты и числа, нужного для стохастического определения поведения пути при столкновении с поверхностью) определяют всё дальнейшее поведение пути. То есть, первые три числа наиболее важны и оказывают максимальное влияние на дальнейшее поведение. В большинстве случаев, генератор, дающий хорошее распределение на малых выборках достаточно быстро выходит из вырожденных состояний, но этого в данном случае недостаточно. Так как ГПСЧ инициализируется заново для каждого трассируемого пути то, что могло бы быть однократным и крайне малозаметным изъяном, будет проявляться самым очевидным образом, искажая результирующее изображение артефактами и не позволяя методу Монте-Карло сходиться к ответу. Так как это требование исключительно строго и очень специфично, не следует отсекать ГПСЧ в процессе выбора на базе этого требования, т.е. если они не могут выйти из вырожденного состояния на первой же итерации. Вместо этого данный вопрос необходимо решать на системном уровне посредством исключения вырожденных зерен для инициализации, ручной инициализации внутреннего состояния на заранее подготовленное состояние с высокой энтропией битов, либо «встряской» ГПСЧ посредством нескольких его вызовов «вхолостую» после инициализации внутреннего состояния. Не рекомендуется использовать для инициализации напрямую целочисленные

координаты вычисляемого пикселя, так как они содержат большое количество нулевых битов, потенциально провоцируя попадание ГПСЧ в вырожденное состояние.

Таким образом, если принять к учёту специфику рендеринга методом трассировки пути, можно сформулировать следующие требования:

– ГПСЧ должен работать быстро;

– внутреннее состояние ГПСЧ должно быть небольшим, желательно кратным размеру регистра;

– ГПСЧ должен возвращать равномерно распределенные числа, желательно, чтобы это свойство соблюдалось на малых выборках;

– требуется очень быстрый выход из вырожденных состояний, их полное отсутствие, или же применение смягчающих мер.

Для контраста следует указать, что для задачи трассировки пути не обязательна непредсказуемость генератора в криптографическом смысле, достаточно лишь комбинации равномерного распределения, приемлемой длины периода и правильной инициализации для того, чтобы избегать артефактов в изображении, поэтому задача выбора ГПСЧ сводится к подбору оптимальной комбинации данных параметров и скорости работы алгоритма.

V. Заключение

В данной статье на основе анализа предлагается набор требований, дающий возможность наиболее эффективно подобрать генератор псевдослучайных чисел для использования в программной реализации рендеринга методом трассировки пути. Рассматриваются как общие требования, позволяющие отсеять неподходящие генераторы в целом, так и специфические требования и рекомендации, возникающие из-за особенностей типовой программной реализации алгоритма. Поясняется, почему часто рекомендуемое решение использования для целей рендеринга криптографических генераторов псевдослучайных чисел является непродуктивным, и каким образом их применение негативно сказывается на скорости работы программы.

Дальнейшее рассмотрение данного вопроса подразумевает проведение сравнительного анализа различных ГПСЧ для конкретизации и уточнения требований и рекомендаций.

Библиография

[1] Kajiya J. T. The rendering equation// Proceedings of the 13th annual conference on Computer graphics and interactive techniques, ACM. 1986. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.63.1402&rep=rep1&type=pdf (Дата обращения: 15.07.2017).

[2] Philip Dutré. Global Illumination Compendium// Computer Graphics, Department of Computer Science, Katholieke Universiteit Leuven, September 29, 2003. URL: https://people.cs.kuleuven.be/~philip.dutre/GI/TotalCompendium.pdf (Дата обращения: 15.07.2017).

Choosing pseudorandom number generator for use with path tracing

I.Y. Sesin, V.V. Nechaev

***Abstract* — This article covers a set of requirements that allow for picking pseudorandom number generator (PRNG) best suited for use in implementations of a rendering method known as path tracing. The set of requirements and recommendations is aimed towards this particular application, and tries to outline issues specific to path tracing as well as avoiding generic problems with PRNGs.**

***Keywords* — pseudorandom number generator, rendering, path tracing.**