DOI: 10.17117/na.2016.04.03.182

Поступила (Received): 25.04.2016

<http://ucom.ru/doc/na.2016.04.03.182.pdf>**Сесин И.Ю., Холкин И.И.****Проблематика использования чисел с плавающей точкой в задачах рендеринга****Sesin I.Yu., Kholkin I.I.****Issues with using floating point numbers in rendering**

В статье производится обзор ряда особенностей работы с типом данных чисел с плавающей точкой в курсе решения задач рендеринга. В их число входят потеря точности, спец. значения, неожиданное поведение кода. Рассматриваются методы предотвращения или противодействия подобным проблемам

Ключевые слова: рендеринг, числа с плавающей точкой, трассировка лучей

Article covers number of issues regarding using floating point number datatype in rendering computations. General overview includes loss of precision, special values and one case of unexpected code behavior. Prevention and mitigation of these issues is discussed

Key words: rendering, floating point number, ray tracing

Сесин Игорь Юрьевич

Бакалавр, магистрант

Московский технологический университет
г. Москва, проспект Вернадского, 78**Sesin Igor Yurievich**

Bachelor, master

Moscow technological university
Moscow, Vernadskogo ave., 78**Холкин Игорь Иванович**

Кандидат технических наук, доцент

Московский технологический университет
г. Москва, проспект Вернадского, 78**Kholkin Igor Ivanovich**Candidate of Engineering Sciences, Associate
ProfessorMoscow technological university
Moscow, Vernadskogo ave., 78

Рендеринг (англ. rendering – «визуализация») в терминах компьютерной графики – процесс получения растрового изображения, исходя из некоторого модельного описания визуализируемых объектов с использованием ЭВМ для выполнения ряда необходимых расчетов.

Для проведения этих расчетов требуется применение типа данных, способного хранить вещественные числа с достаточной точностью. Исторически, широкое распространение получил тип данных числа с плавающей точкой, вытеснив числа с фиксированной точкой в расчетах общего назначения. Существует много форм записи чисел с плавающей точкой, но в настоящее время, наиболее часто встречаемой репрезентацией подобных чисел в вычислительной технике является репрезентация, утвержденная стандартом IEEE 754.

Однако, применение такого типа данных на практике для выполнения точных расчетов, требуемых в классе задач рендеринга, приводит к столкнове-

нию с рядом особенностей, влияющих, главным образом, на точность результата. Эти особенности, возникают вследствие того, что числа с плавающей точкой не могут адекватно представлять всё множество вещественных чисел.

Эти особенности проявляются только на этапе построения программного средства, в ходе тестирования и отладки, и они не следуют очевидным образом из самого мат. аппарата рендеринга. Часть из этих особенностей можно избежать, часть – только смягчить, но не убрать полностью. Рассмотрим их.

Вообще говоря, при любых расчетах с числами с плавающей точкой существует некоторая ошибка. Эта ошибка называется ошибкой округления и появляется вследствие того, что далеко не все числа можно отобразить в такой записи. Для подобных чисел берется ближайшее число, которое может быть отображено в данном формате. Обычно эта ошибка настолько мала, что её можно вообще не учитывать, поскольку она практически не влияет на результат. Однако существует ряд ситуаций, в которых провоцируется появление серьезных ошибок, могущих повлиять на весь расчет в целом. В зависимости от участка кода рендерера, в котором происходят ошибки округления чисел с плавающей точкой, могут наблюдаться самые различные результаты в полученном изображении, от зашумления изображения («эффект снега») до изменения позиций объектов.

Одной из подобных ситуаций является ситуация так называемого катастрофического исключения (англ. «Catastrophic Cancellation» [1]), когда происходит вычитание двух близко расположенных чисел. Это приводит к тому, что «исключается» большое количество значащих цифр, что приводит к появлению существенной относительной ошибки. Это происходит чаще всего в случаях, когда в этих числах уже есть биты, подверженные ошибке округления, как правило, в результате выполнения действий над ними. Можно избежать этого, реструктурировав формулу [1].

Так же возможно накопление ошибок. Причины, по которым малые ошибки округления могут накопиться во влияющие на результат работы программы, разнятся. Ошибки округления могут накапливаться в ходе долгих итеративных расчетов (особенно, если вовлечено последовательное суммирование), или округления промежуточных расчетов, в результате использования недостаточно точных типов данных для хранения больших чисел, а также из-за неудачных оптимизаций на уровне компилятора [2].

Для того чтобы предотвратить накопление ошибок чаще всего используется увеличение точности переменных, в которых возникает ошибка. Этот метод достаточно эффективен, но он – далеко не панацея. Важно так же позаботиться о том, чтобы абсолютно все промежуточные данные не округлялись, в том числе и неявно, посредством хранения в переменных меньшей точности. Важно избегать совершения итеративно повторяющихся действий с одними и теми же переменными, особенно векторами. Иногда будет более точно высчитать значение заново, нежели вводить корректировки для старого, «кэшированного» значения. Например, вместо того чтобы хранить некоторый вектор, который будет далее вращаться на фиксированные углы, следует хранить угол и

длину вектора, из которых будет каждый раз высчитываться новый вектор. Такой подход очень важен при работе с нормализованными векторами, так как длина нормализованного вектора равна единице, а при последовательных операциях над одним вектором, в результате наложения множества ошибок, его длина может измениться, что, в свою очередь уже вызовет проблемы в дальнейшем коде, законно полагаящем, что он оперирует с нормализованным вектором.

Так же в стандарте указываются несколько специальных значений, которые отображают специальные случаи чисел. Наиболее проблематичными из них являются NaN и $\pm\text{inf}$.

Спец. значение NaN (англ. «Not a Number» – Не Число) обозначает проведение математически некорректной операции, такой как, например, деления нуля на нуль. В зависимости от компилятора и среды работа программы может либо продолжиться, либо быть остановлена при получении такого результата.

Важно, что большинство операций (но не все!) с NaN в результате дадут NaN, и это ошибочное значение может легко распространяться по цепочке вычислений, значительно усложняя поиск места возникновения ошибки.

Без специальных проверок, это может приводить к совершенно непредсказуемым последствиям работы рендерера, от банального исчезновения объектов с «испорченными» координатами до краха программы.

Согласно стандарту, значение NaN должно быть неравно любому числу, включая само себя. Таким образом, проверка на равенство переменной самой себе позволяет выявить возникновения NaN в процессе расчетов.

Бесконечности $+\text{inf}$ и $-\text{inf}$ образуются в результате деления чисел на нуль, а так же при выходе за числовой диапазон. Они так же будут распространяться по цепочке вычислений, в некоторых случаях образуя NaN.

Следует учитывать, что можно выйти за диапазон значений, поделив не только на нуль, но и на достаточно близкое к нему число, поэтому при делении, проверять делитель следует не на равенство нулю, а на принадлежность участку числовой прямой, близкому к нулю.

Другой особенностью является неявная работа с floating point регистром FPU. Дело в том, что для непосредственного выполнения вычислений в FPU используется регистр длиной в 80 бит. При сохранении в переменную, результат будет обрезан по длине переменной.

Но далеко не всегда скомпилированный код записывает значение в переменную, прежде чем начинает с ним работать. Это является положительным моментом с точки зрения оптимизации, но может привести к редким ситуациям, когда код работает не так, как ожидается.

Существуют языковые конструкции, которые провоцируют подобное поведение, например, сравнение значения переменной с некоторым вычисленным «на месте» значением. Так как значение переменной уже было обрезано по границе, допустим, в 64 бита, при сравнении со значением, содержащимся в 80-битном регистре и имеющим дополнительные биты точности, будет установлено их неравенство. Исключением является редкий случай, когда дополнительные биты равны 0.

Учёт рассмотренных особенностей применения чисел с плавающей точкой позволяет предупредить появление проблем в процессе программной реализации рендереров и высокоточного программного обеспечения в целом.

Список используемых источников:

1. David Goldberg. *What Every Computer Scientist Should Know About Floating-Point Arithmetic*. March, 1991.
URL: http://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html
2. Kahan W. *How Futile are Mindless Assessments of Roundoff in Floating-Point Computation?*
URL: <https://www.cs.berkeley.edu/~wkahan/Mindless.pdf>
3. Kahan W. *IEEE Standard 754 for Binary Floating-Point Arithmetic*.
URL: <https://www.cs.berkeley.edu/~wkahan/ieee754status/IEEE754.PDF>

© 2016, Сесин И.Ю., Холкин И.И.

Проблематика использования чисел с плавающей точкой в задачах рендеринга

© 2016, Sesin I.Yu., Kholkin I.I.

Issues with using floating point numbers in rendering