УДК 004.92

О МЕТОДЕ ОТСЕЧЕНИЯ ПРИ РАСТЕРИЗАЦИИ СЛОЖНЫХ ОБЪЕКТОВ

С. А. Карпухин[1]

Предложен новый метод отсечения при растеризации сложных объектов с помощью вычисления одной точки. Дана оценка количества требуемых операций в сравнении с известными методами. Показано, что при решении задач вычислительной геометрии методами машинной графики этот метод дает существенный прирост производительности.

Ключевые слова: растеризация, отсечение, методы машинной графики.

1. Введение. Компьютерная графика за последние 50 лет стала очень продуктивным и активно развивающимся направлением вычислительной математики. В частности, запросы интерактивных мультимедийных приложений привели к разработке высокоэффективных алгоритмов отрисовки (растеризации) различных объектов. Подобная успешность алгоритмов и аппаратная реализация многих из них привели к тому, что методы компьютерной графики начали применяться для решения вычислительных задач [2, 3, 5]. Однако специфика вычислительных задач предъявляет новые требования к алгоритмам компьютерной графики. Например, в [3] предлагается улучшать точность приближенной диаграммы Вороного путем ее отрисовки с бóльшим разрешением вокруг вершин. Это приводит к задаче отсечения объектов большого размера малым окном, достаточно редкой в мультимедийных приложения компьютерной графики. В настоящей статье предлагается новый подход к решению этой задачи и демонстрируются его преимущества на примере отсечения и растеризации окружности.

Сама задача отсечения возникла одновременно с задачей отрисовки и формулируется следующим образом: дан некоторый геометрический объект и область пространства, называемая окном. Внутри окна задан растр, т.е. отображение конечного множества точек в некоторое конечное множество атрибутов, например цветов. Требуется отобразить на растре часть геометрического объекта, лежащую внутри окна.

Известные на текущий момент методы отсечения в компьютерной графике представлены двумя классами: аналитические алгоритмы и алгоритм поточечного теста. *Аналитические алгоритмы* полностью вычисляют геометрию частей объекта, лежащих внутри окна, а затем растеризуют полученные геометрии. К этому классу относится широко известный алгоритм Сазерлэнда–Коэна отсечения отрезков [1, 6] и его последующие модификации. *Алгоритм поточечного теста* отображает всю геометрию исходного объекта на расширенный растр, включающий в себя растр окна, а затем для каждой точки растрового представления объекта проверяет, лежит ли она внутри окна [1]. Такой алгоритм реализован в большинстве современных графических библиотек, а также аппаратно в графических процессорах.

Алгоритм поточечного теста не применим к обозначенной выше задаче, так как требует значительных вычислительных затрат на отрисовку точек объекта, заведомо лежащих вне окна [1]. Аналитические алгоритмы хорошо работают для отрезка прямой, но могут быть недостаточно эффективны в случае аналитически сложных объектов, как это будет показано далее.

2. Идея нового метода отсечения. Основа нового метода состоит в аналитическом вычислении одной точки объекта, лежащей внутри окна или в его окрестности, и в последовательном построении объекта от этой точки с проверкой принадлежности новых точек окну. Такой подход объединяет преимущества известных методов и устраняет основные их недостатки: вычисление одной точки в окрестности окна, скорее всего, реализуется проще полного описания отсеченной части, а проверку принадлежности новых точек окну можно совместить с условием остановки типовых циклов растеризации, благодаря чему бóльшая часть растрового представления объекта вычисляться не будет. Алгоритм растеризации при замене условия остановки замедляется незначительно и, как показывают практические испытания, не устраняет выигрыша нового метода отсечения.

Для наглядной иллюстрации преимуществ нового подхода применим его к задаче отсечения окружности, так как окружность — достаточно сложный объект, чтобы увидеть превосходство над стандартными аналитическими методами. Кроме того, окружность и сфера являются основными образующими фигурами для алгоритмов геометрической оптимизации по евклидовой метрике, подобных [2, 5], а обобщение

[1] Московский государственный университет им. М. В. Ломоносова, механико-математический факультет, Ленинские горы, д. 1, 119991, Москва; аспирант, e-mail: ks-linp@yandex.ru

алгоритма на случай сферы не представляет трудностей. В качестве окна возьмем прямоугольник со сторонами, параллельными осям координат. В трехмерном случае ему будут соответствовать прямоугольный параллелепипед, другие типы окон практически не встречаются на практике и в исследованиях.

В соответствии с изложенной идеей метод отсечения при растеризации заключается в поиске стартовой точки — точки окружности, лежащей в окрестности прямоугольника (окна), и в последующей растеризации части окружности начиная с этой точки (рис. 1). При этом ограничиваемся случаем, когда внутри окна находится не более двух частей окружности. В остальных случаях размеры окружности того же порядка, что и размер растра, и можно применить алгоритм поточечного теста.

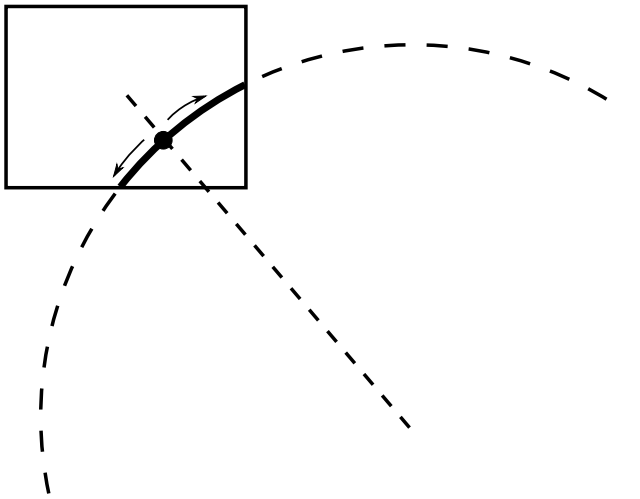

Рис. 1. Метод одной точки

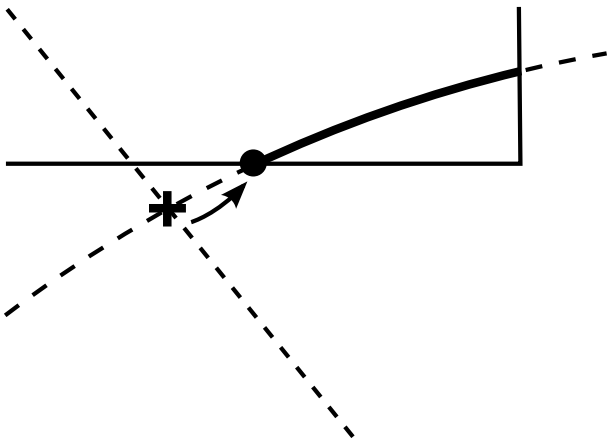

Рис. 2. Стартовая точка вне окна

Стартовая точка выбирается как точка окружности, лежащая на прямой, соединяющей центр окружности и центр окна. Формулы расчета следующие:

$$\Delta x = x_C^b - x_C^c, \quad \Delta y = y_C^b - y_C^c,$$
$$d = \sqrt{\Delta x^2 + \Delta y^2}, \quad x_s = \frac{\Delta x\, R}{d}, \quad y_s = \frac{\Delta y\, R}{d},$$

где x_C^b, y_C^b — координаты центра окна, x_C^c, y_C^c — координаты центра окружности и R — радиус окружности.

Заметим, что возможен случай, когда стартовая точка находится вне окна. В этом случае можно воспользоваться процедурой растеризации и построить без отображения на растре часть окружности до пересечения с окном, после этого за стартовую точку принимается первая точка окружности, лежащая внутри окна (рис. 2). При этом количество итераций растеризации поиска стартовой точки и отрисовки в сумме не будет превышать длины растра вдоль соответствующей оси, что эквивалентно средним затратам на отрисовку отсеченной части окружности.

В качестве алгоритма растеризации воспользуемся общепринятым алгоритмом средней точки, называемым также алгоритмом Брезенхема [1, 4]. Работа алгоритма заключается в вычислении следующей точки окружности до тех пор, пока не выполнится условие остановки. Условием остановки обычно является достижение конечной точки. Заменив его на условие принадлежности окну, получим алгоритм, растеризующий дугу окружности, лежащую внутри окна. При этом следует помнить, что стартовая точка почти всегда является для требуемой дуги строго внутренней и необходимо выполнять растеризацию в двух направлениях от стартовой точки.

3. Оценка количества операций. Оценим количество операций нового алгоритма в сравнении с известными. Применительно к окружности, это — стандартный алгоритм поточечного теста [1] и быстрый алгоритм [7].

Алгоритм поточечного теста не требует дополнительных затрат на отсечение, однако проводит тест для каждой точки. На окружностях малых радиусов, т.е. сравнимых с размерами окна, он работает быстрее аналитических, но при достаточно большом радиусе окружности становится практически неприменимым.

Новый алгоритм отсечения в большинстве случаев растеризует те же точки, что и быстрый алгоритм, поэтому необходимо сравнить только их затраты на отсечение. При этом простой тест на пересечение

границ окружности с экраном проводится в обоих алгоритмах одинаково, поэтому его тоже можно не учитывать.

Для краткости в подсчете будем обозначать операции сравнения через C, целочисленной арифметики через A, вещественной арифметики через R, а вызовы трансцендентных функций через T.

Утверждение 1. *Быстрый алгоритм требует в лучшем случае* $10A + 26C + 10R + 6T$ *операций на отсечение.*

Доказательство. В соответствии с описанием, изложенным в [7], быстрый алгоритм выглядит следующим образом.

1. Находим углы пересечения окружности с прямыми, ограничивающими прямоугольное окно.
2. Сортируем углы по возрастанию.
3. Проходим все полученные угловые отрезки и вычисляем, какие из них соответствуют частям дуги внутри окна.
4. Полученные углы преобразуются в координаты начальной и конечной точек для растеризации.

На первом шаге производится грубая проверка каждой границы на пересечение ($4(A + C)$ операций), затем для каждого найденного пересечения вычисляются угловые параметры точек пересечения, что требует в среднем $2(T + 3R)$ операций, так как при больших радиусах в лучшем случае пересекаются две границы.

Далее сортируется массив из параметров концов дуг — это в лучшем случае два пересечения, на каждом две дуги по две координаты, т.е. 8 вещественных элементов и около $18C$ операций, например для сортировки слиянием.

На следующем шаге массив угловых координат проходится с подсчетом индекса наложений ($2(2C+A)$ операций). Затем вычисляются координаты концов: в лучшем случае это одна дуга, что соответствует двум концам, для каждого из которых требуется $2T + 2R + 2A$ операций на вычисление координат, всего $4T + 4R + 4A$ операций.

В сумме по всем этапам получаем указанную оценку. Что и требовалось доказать.

Утверждение 2. *Новый алгоритм требует в худшем случае* $19A + 12C + 4R + 1T$ *операций.*

Доказательство. Вычисление стартовой точки по выписанным выше формулам требует $5A+1T+4R$ операций. Разбор случаев, в которых стартовая точка находится вне окна, требует $12C + 14A$ операций (для существующей реализации в худшем случае). Утверждение 2 доказано.

Поскольку арифметические операции выполняются быстрее сравнений, вещественных и трансцендентных операций, новый алгоритм уступает быстрому только по арифметическим операциям и общее количество операций в новом алгоритме меньше, то можно сделать вывод, что в любом случае новый алгоритм эффективнее быстрого. На практике это различие будет намного существеннее, так как оценки брались для быстрого алгоритма наилучшие, а для нового — наихудшие. Кроме того, новый алгоритм выполняет значительно меньше трансцендентных операций и операций сравнения, являющихся наиболее медленными. Приведенное ниже сравнение времени работы рассмотренных реализаций подтверждает эти выводы.

Таблица 1

Размер растра 64×64, 5 000 000 окружностей

Алгоритм	Окр. малого радиуса	Окр. большого радиуса
Поточечный тест	5.5 с	33.8 с
Быстрый алгоритм	7.4 с	7.4 с
Новый алгоритм	5.3 с	5.3 с

Таблица 2

Размер растра 1000×1000, 1 000 000 окружностей

Алгоритм	Окр. малого радиуса	Окр. большого радиуса
Поточечный тест	18.5 с	610 с
Быстрый алгоритм	9.5 с	9.5 с
Новый алгоритм	8.5 с	8.5 с

4. Практические результаты и выводы. Сравнение проводилось путем отрисовки окружностей малого (порядка размера окна) и большого (на два порядка больше размера окна) радиуса. Сравнение производилось для двух растров: малого (64×64), типичного для геометрических методов решения численных задач, и большого (1000×1000), типичного для задач компьютерной графики. Время работы измерялось для реализаций на CPU Intel Core I5 2GHz в один поток. Результаты представлены в табл. 1 и 2.

Меньший выигрыш производительности на большом растре объясняется тем, что увеличилась часть

времени, приходящаяся на растеризацию при сохранении затрат на отсечение в сравнении с малым растром, а алгоритмы растеризации обоих методов совпадают с точностью до условия остановки.

Таким образом, в настоящей статье был предложен новый метод отсечения сложных объектов экраном при растеризации. На примере растеризации окружности показан значительный (до 40%) прирост производительности при растеризации с отсечением больших объектов. Алгоритм требует модификации стандартных процедур растеризации, но тем не менее весьма эффективен в графических методах решения геометрических и оптимизационных вычислительных задач, так как согласно результатам профилирования, проведенного при реализации одного из таких алгоритмов, бóльшая часть времени (до 90%) тратится на отсечение и растеризацию.

Следует также заметить, что современные многопроцессорные GPU не уменьшают практической ценности алгоритма, поскольку графические методы решения вычислительных задач, как правило, требуют отрисовки огромного числа окружностей большого (по сравнению с размерами окна) радиуса. Напротив, простота реализации нового алгоритма позволяет перенести его на GPU достаточно эффективно.

СПИСОК ЛИТЕРАТУРЫ

1. *Иванов Д., Карпов А., Кузьмин Е.* и др. Алгоритмические основы растровой машинной графики. М.: БИНОМ, 2007.
2. *Denny M.* Solving geometric optimization problems using graphics hardware // Computer Graphics Forum. 2003. **22**, N 3. 441–451.
3. *Hoff K.E., Keyser J., Lin M., et al.* Fast computation of generalized Voronoi diagrams using graphics hardware // Proc. of the 26th Annual Conference on Computer Graphics and Interactive Techniques. New York: ACM Press, 1999. 277–286.
4. *Pitteway M.L.V.* Algorithm for drawing ellipses or hyperbolae with a digital plotter // The Computer Journal. 1967. **10**. 282–289.
5. *Rong G., Tan T.-S., Cao T.-T., Stephanus I.* Computing two-dimensional Delaunay triangulation using graphics hardware // Proc. of the 2008 Symposium on Interactive 3D Graphics and Games. New York: ACM Press, 2008. 89–97.
6. *Sproull R., Sutherland I.* A clipping divider // Proc. of the Fall Joint Computer Conference. Washington: Thompson Books, 1968. 765–775.
7. *Srinivasan R.V.* A fast circle clipping algorithm // Graphics Gems III / Ed. by D. Kirk. San Diego: Academic, 1992. 182–187.

Поступила в редакцию
23.07.2013