Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

АЛГОРИТМ ВАТТИ ОТСЕЧЕНИЯ МНОГОУГОЛЬНИКОВ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Петруниной Ирины Николаевны

Научный руководитель
зав. кафедрой, к. ф.-м. н. _____ С. В. Миронов

Заведующий кафедрой
к.ф.-м.н. _____ С. В. Миронов

Саратов 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью бакалаврской работы является систематизация сведений о реализации алгоритмов двумерного отсечения многоугольников методом Ватти и методом Вейлера—Азертона, а также реализация алгоритма Ватти в среде Microsoft Visual Studio Community 2015.

Актуальность данной работы заключается в том, что на данный момент нет полного и точного источника, где собраны данные о реализации алгоритмов Вейлера—Азертона и Ватти, ни на русском, ни на английском языке. А использование алгоритма Ватти приведет к оптимизации работы приложений в области компьютерной графики, т.к. данный алгоритм является одним из самых быстрых и эффективных на сегодняшний день [1].

Алгоритм Ватти может обрабатывать любые многоугольники (выпуклые, невыпуклые, с самопересечениями и отверстиями), алгоритм Вейлера—Азертона не поддерживает работу с многоугольниками, которые имеют самопересечения. Оба алгоритма могут отсекать по произвольному окну, что значительно расширяет область применения данных алгоритмов. Также оба алгоритма могут быть модифицированы для работы с логическими операциями над многоугольниками, а алгоритм Ватти предусматривает и работу с трапециями.

Данная дипломная работа состоит из пяти глав. В первой главе «Алгоритм отсечения многоугольников методом Вейлера—Азертона» [2–5] описаны задачи алгоритма, представление многоугольников, описание алгоритма, его достоинства и недостатки. Во второй главе «Алгоритм отсечения многоугольников методом Ватти» [6–8] содержится информация о задачах алгоритма, представлении многоугольников, описании алгоритма, его достоинствах и недостатках. Кроме этого, в этой главе рассмотрены возможные модификации алгоритма Ватти. В третьей главе «Сравнение алгоритмов Вейлера—Азертона и Ватти» приведено сравнение рассмотренных алгоритмов по трудоемкости, типу исходных многоугольников, возможным ограничениям и т.д. Четвертая глава «Реализация алгоритма Ватти» содержит информацию о формате входных данных, о том, что должно быть реализовано в ходе выполнения бакалаврской работы, условиях реализации и т.д. В последней главе «Создание приложения» рассматривается реализация приложения, описываются конкретные классы и методы проекта, а также приводится результат работы.

1 Алгоритм Ватти

Алгоритм Ватти — это общий алгоритм отсечения многоугольников по произвольной области видимости. Данный алгоритм является одним из самых эффективных. Он предназначен для отсечения произвольных многоугольников по любым отсекателям (выпуклым или невыпуклым). Алгоритм может обрабатывать сложные многоугольники, которые имеют самопересечения, и многоугольники с отверстиями.

Алгоритм Ватти также называется скан-линейным. Его основная идея [7] состоит в том, что все вершины аргументов упорядочиваются по значению их ординат. Далее каждая из этих вершин порождает горизонтальную скан-линию. При последовательном просмотре скан-линий производится анализ их пересечений с точками и рёбрами аргументов, поддерживается список контуров, в который вносятся изменения, соответствующие характеру и последовательности пересечения очередной скан-линии с вершинами и сторонами.

В алгоритме отсечения Ватти многоугольники представляются через набор левой и правой границ, являющихся связанными списками левых и правых ребер соответственно, которые образуют пары. Каждая из этих границ начинается в локальном минимуме многоугольника и заканчивается в локальном максимуме.

Первым шагом алгоритма является определение левых и правых границ отсекателя и отсекаемых многоугольников. Эту информацию необходимо добавить в список локальных минимумов (СЛМ). Этот список состоит из списка составленных пар лево-правых границ. СЛМ отсортирован в порядке возрастания координаты Y соответствующего локального минимума. Необходимость в этом отпадает, если начальные горизонтальные ребра находятся в левой или правой границе.

Границы в СЛМ определены так, что ребра, входящие в них, либо все левые, либо все правые. Тем не менее, удобнее иметь более общее понятие левой или правой границы. Таким образом, в дальнейшем левая или правая граница будет обозначать любую связанную последовательность ребер, у которой только первое ребро должно быть соответственно левым или правым. По-прежнему будем считать, что граница начинается в локальном минимуме и заканчивается в локальном максимуме.

Затем за несколько шагов необходимо построить выходные многоуголь-

ники из последовательностей «неполных» многоугольников, каждый из которых является «V-образным» списком, состоящим из вершин, расположенных на левой стороне, поступающих из левой границы, и на правой стороне, поступающих с правой границы, и где существует общая вершина, то есть вершина, расположенная у основания «V», которая и является локальным минимумом. Обозначим через $P[p_0p_1...p_n]$ неполный многоугольник с вершинами p_{i_0}, p_{i_1}, $\cdots$, p_{i_n}, где p_{i_0} — первая точка, а p_{i_n} — последняя. Точки p_{i_0} и p_{i_n} находятся на вершине неполной левой и правой границы соответственно. Пусть некоторая вершина p_{i_m} является вершиной локального минимума, которая соединяет две границы, но, так как она не будет использоваться, нет необходимости указывать m в обозначениях. Заметим, что ребра левой и правой границ не всегда являются правыми или левыми внутренними сторонами многоугольника. Если построение многоугольника завершено, то p_{i_0} и p_{i_n} — одна и та же вершина, являющаяся локальным максимумом, но на всех остальных промежуточных шагах построения многоугольника вершины p_{i_0} и p_{i_n} различны. Тем не менее, p_{i_0} и p_{i_n} всегда будут соответствовать началу вершин текущих левых и правых неполных границ соответственно. Хороший способ реализации таких неполных многоугольников — это реализация с помощью кругового связанного списка, или цикла, и указателя, который указывает на последний элемент в списке.

Алгоритм вычисляет границы выходных многоугольников из СЛМ путем сканирования всей области от основания до вершины с помощью так называемых скан-полос. Скан-полосы — это участки между двумя горизонтальными линиями сканирования (не обязательно смежными), причем каждая из этих линий содержит, по крайней мере, одну вершину многоугольников и не содержит вершин между ними. Следует отметить, что линии сканирования, которые определяют скан-полосы, вычисляются не сразу, а постепенно снизу вверх. Информация о скан-полосах хранится в списке скан-линий (ССЛ), который является упорядоченным списком из ординат всех линий сканирования, определяющих скан-полосы. Этот перечень возрастающих значений будет рассматриваться как стек. Так как сканируется вся область, необходимо создать активный список ребер (АСР), который является упорядоченным списком, состоящим из всех ребер, пересекающих текущую скан-полосу.

При обработке скан-полосы сначала необходимо проверить СЛМ, чтобы

увидеть начинаются ли его граничные пары в нижней части скан-полосы. Эти границы соответствуют локальным минимумам и могут начать новый выходной многоугольник или разбить один на два в зависимости от того, начинается локальный минимум с лево-правой или право-левой пары ребра. Затем любые новые ребра из СЛМ добавляются в АСР, который необходимо проверить на наличие пересечений ребер в пределах скан-полосы. Эти пересечения влияют на выходные многоугольники и рассматриваются отдельно в первую очередь. В конце происходит обработка ребер в АСР.

Некоторые ребра и вершины, которые встречаются в выходных многоугольниках или создаются для них, будут принадлежать границам отсекателя, другие не будут. Назовем вершину (ребро) способствующей или неспособствующей вершиной (ребром) в зависимости от того, принадлежит ли вершина (ребро) выходным многоугольникам. Если вершина не является локальным минимумом или максимумом, то она будет называться левой или правой промежуточной вершиной в зависимости от того, принадлежит ли она к левой или правой границе соответственно. Так как работа алгоритма основывается на встречающихся вершинах, то алгоритм в основном сводится к тщательному анализу трех случаев:

1. Вершина является локальным минимумом;
2. Вершина является левой или правой промежуточной вершиной;
3. Вершина является локальным максимумом.

Локальные минимумы встречаются, когда элементы из СЛМ становятся активными. Промежуточные вершины и локальные максимумы встречаются при сканировании АСР. Пересечения ребер также могут привести к этим трем случаям.

Рассмотрим, что нужно делать с пересечениями ребер внутри скан-полосы. Способ, с помощью которого эти пересечения обрабатываются, зависит от того, имеются одинаковые или разные ребра. Пересечения считаются одинаковыми, только если оба ребра способствующие, и в этом случае точка пересечения должна рассматриваться и как левая, и как правая промежуточная вершина. При одинаковых пересечениях, если одно ребро способствующее, то и другое ребро также должно быть способствующим. Разные пересечения всегда должны быть обработаны. Обработка точки их пересечения зависит от их типа, боковой стороны и относительной позиции в АСР.

Алгоритм может быть оптимизирован в общем случае прямоугольных границ отсечения. Другая оптимизация возможна, если отсекатель фиксируется (прямоугольный или нет) путем единственного вычисления его границ и инициализации СЛМ этих границ в начале вызова алгоритма отсечения.

Особенностью алгоритма Ватти является то, что он может быть легко модифицирован для генерации трапеции. Это особенно удобно для сканирования линейно-ориентированных алгоритмов визуализации. Каждый локальный минимум начинает трапецию или разбивает существующую на две в зависимости от того, начинается локальный минимум с лево-правой (способствующий случай) или право-левой (неспособствующий случай) пары ребра. В способствующем локальном минимуме мы создаем настройки трапеции для всего ее поля, кроме Y-координаты верхней вершины. Трапеции выводят локальные максимумы и левые или правые промежуточные вершины. Неспособствующий локальный минимум должен вывести трапецию, которую он собирается разбить, а ее указатели соответствующих ребер обновляются для двух новых трапеций.

Также можно использовать алгоритм Ватти не только для пересечения, но и для других операций. Например, для объединения или пересечения многоугольников.

Главным достоинством алгоритма Ватти является скорость работы. Данный алгоритм является одним из самых быстрых и эффективных на сегодняшний день [1].

Еще одним плюсом алгоритма является представление многоугольников через наборы левой и правой границ, являющихся связанными списками левых и правых ребер соответственно, которые образуют пары. Это упрощает восприятие каждого отсекаемого многоугольника и упрощает работу с ними. Также к достоинствам стоит отнести следующее:

- Возможность работы с трапециями;
- Использование не только прямоугольного, но и выпуклого и невыпуклого отсекателя;
- Использование как выпуклого, так и невыпуклого отсекаемого многоугольника, а также многоугольника с отверстиями;
- Возможность использования алгоритма для всех логических операций (разность, объединение, пересечение, xor и др.);

— Обработка самопересечений.

К недостаткам стоит отнести сложность реализации алгоритма и большое количество крайних случаев, возникающих при реализации.

2 Сравнение алгоритмов Ватти и Вейлера—Азертона

Единственный алгоритм, который можно сравнить с алгоритмом Ватти, — это алгоритм Вейлера—Азертона. Оба алгоритма могут обрабатывать выпуклые, невыпуклые многоугольники, а также многоугольники с отверстиями. Но алгоритм Вейлера—Азертона не предусматривает работу со сложными многоугольниками, которые имеют самопересечения. Также исходными данными в алгоритме Вейлера—Азертона могут быть только регулярные многоугольники, имеющие произвольное количество контуров. В алгоритме Ватти таких ограничений нет. Кроме того, недостатком алгоритма Вейлера—Азертона является ограничение на вид исходных данных.

Оба алгоритма имеют достаточно сложную реализацию. Однако у алгоритма Ватти скорость работы значительно выше, чем у алгоритма Вейлера—Азертона. Самым важным недостатком этих алгоритмов является необходимость учета большого количества крайних случаев, возникающих при реализации, и, как следствие, низкая вычислительная устойчивость, проявляющаяся при поиске точек пересечения сторон многоугольников. Алгоритмы имеют достаточную высокую трудоемкость (см. табл. 1) [9].

В таблице 1 приведены сравнительные характеристики данных алгоритмов. В таблице использованы следующие обозначения:

1. n_1 — количество вершин первого многоугольника;
2. n_2 — количество вершин второго многоугольника;
3. n_0 — количество новых точек пересечения сторон двух многоугольников;
4. k — общее количество контуров в многоугольниках;
5. $n = n_1 + n_2 + n_0$.

Таблица 1 – Сравнительные характеристики алгоритмов Вейлера—Азертона и Ватти

	Алгоритм Вейлера—Азертона	Алгоритм Ватти
Тип исходных многоугольников	Простые с ориентированными контурами	Общие
Многоконтурность	+	+
Вычислительная устойчивость	−	−
Трудоемкость	$O(n \cdot \log(n_1 + n_2))$	$O(n \cdot \log(n))$

3 Практическая часть. Реализация алгоритма Ватти

В ходе выполнения работы было реализовано приложение в среде `Visual Studio` с помощью `Windows Forms` — интерфейса программирования приложений (API), отвечающего за графический интерфейс пользователя.

В данной работе был реализован алгоритм отсечения многоугольников методом Ватти. Программа отсекает как выпуклые, так и невыпуклые многоугольники по области видимости, которая может быть также выпуклым или невыпуклым многоугольником. Отсекатель фиксируется после первой отрисовки, а отсекаемые многоугольники можно перемещать по области внутри отсекателя (вправо, влево, вверх, вниз). Отсекаемых многоугольников может быть несколько. Также может быть отрисована прямоугольная рамка, которая может отсутствовать. Ее стороны параллельны экрану. Если такая рамка введена во входном файле, то область видимости отсекается по этой рамке. Если не введен отсекатель, то областью видимости является прямоугольная рамка. Во входном файле обязательно должен присутствовать отсекатель или рамка. Если нет ни отсекателя, ни рамки, то входные данные считаются некорректными.

Используется текстовый формат входного файла. Входной файл может содержать строки с командами:

$frame\ Vcx\ Vcy\ Vx\ Vy$ — команда выделения кадра;

$polygon\ n\ x_1\ y_1\ \cdots\ x_n\ y_n$ — задается многоугольник с n вершинами, в котором числа x_i или y_i — координаты i-ой вершины многоугольника в порядке обхода вершин по часовой стрелке;

$clip\ n\ x_1\ y_1\ x_2\ y_2\ \cdots\ x_n\ y_n$ — задается область видимости — выпуклый многоугольник с n вершинами, в котором числа x_i или y_i — координаты i-ой вершины многоугольника в порядке обхода вершин по часовой стрелке.

ЗАКЛЮЧЕНИЕ

В данной дипломной работе были выполнены все поставленные цели и задачи. Был реализован двумерный алгоритм отсечения многоугольников методом Ватти на языке C++ в среде Microsoft Visual Studio Community 2015, проведено тестирование данного приложения. Были систематизированы сведения о реализации алгоритмов двумерного отсечения методом Вейлера—Азертона и методом Ватти, произведено сравнение этих алгоритмов. Стоит отметить, что на данный момент алгоритм Ватти является одним из самых эффективных алгоритмов. Алгоритм Вейлера—Азертона уступает алгоритму Ватти по многим пунктам.

Рассмотренные алгоритмы применяются для отсечения многоугольников по произвольной области видимости. Алгоритм Ватти — это уникальный алгоритм. Он обладает довольно высокой скоростью работы, а также не генерирует лишние стороны для отсеченного многоугольника, проходящие вдоль ребра отсечения, и может обрабатывать сложные многоугольники с самопересечениями и отверстиями. Также данный алгоритм можно легко модифицировать для работы с логическими операциями и трапециями, что расширяет его область применения. Аналогов этого алгоритма на данный момент не существует.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Peng, Y.* Efficient algorithm for general polygon clipping // Proceedings of The 6th International Conference on Computer-Aided Industrial Design and Conceptual Design 2005. — Netherlands: Delft, Сентябрь 2005. — Pp. 182–185.

2 *Миронов, С. В.* Лекции по компьютерной графике [Электронный ресурс] / С. В. Миронов. — С. 51–93. — URL: `http://course.sgu.ru/file.php/564/Lections/lections.pdf`.

3 *Weiler, K.* Hidden surface removal using polygon area sorting [Электронный ресурс] // Computer Graphics. — Ithaca, New York: 1977. — Pp. 214–222. — URL: `https://www.cs.drexel.edu/~david/Classes/CS430/HWs/p214-weiler.pdf`.

4 *Вельтмандер, П. В.* Машинная графика. Учебное пособие в 3-х книгах. Книга 2. Основные алгоритмы / П. В. Вельтмандер. — Новосибирск: Новосибирский университет, 1997.

5 *Херн, Д.* Компьютерная графика и стандарт OpenGL / Д. Херн, М. П. Бейкер. — Вильямс, 2005.

6 *Agoston, M. K.* Computer Graphics and Geometric Modeling. Implementation and Algorithms / M. K. Agoston. — London: Springer, 2005.

7 *Матвеев, И. А.* Получение оверлеев векторных данных большого объёма // ГрафиКон'2013: 23-я Международная конференция по компьютерной графике и зрению. — Владивосток: Дальнаука, Сентябрь 2013. — Pp. 182–185.

8 *Роджерс, Д.* Алгоритмические основы машинной графики / Д. Роджерс. — Москва: Мир, 1989.

9 *Скворцов, А. В.* Комплексное исследование и разработка эффективных вычислительно устойчивых алгоритмов вычислительной геометрии и их реализация в геоинформационной системе / А. В. Скворцов. — Томск: Томский государственный университет, 2002.