Математически обзор и софтуерна реализация на алгоритъма на Брезенхам за линия

Боян Зарев
Факултет Компютърни системи и технологии
Технически Университет – София
бул. „Климент Охридски“ № 8, 1000 София, България
bzarev@tu-sofia.bg

***Резюме* –Този литературен обзор разглежда алгоритъма на Брезенхам за изчертаване на права линия в растерна среда. В математическия обзор се анализират принципите на алгоритъма. Софтуерната реализация е направена на Python с библиотеката Pygame. Pygame служи единствено за обработка на събития и визуализация на пикселите, докато логиката на алгоритъма е реализирана независимо.**

***Ключови думи* – алгоритъм на Брезенхам; линия; растеризация**

I. Въведение

Софтуерните пакети за графични приложения предоставят библиотека от функции, които се използват за създаване на изображения. Едно от първите неща, които трябва да се направи при създаването на изображение, е да се опишат съставните части на сцената, която ще бъде илюстрирана. Компонентите на картината могат да бъдат дървета, терен, мебели, стени, автомобили и най-различни предмети. Преди създаването на сцената, трябва да се опише структурата на отделните обекти и техните координатни местоположения. Функциите в графичния пакет, които дефинират структурата и местоположенията на обектите се наричат геометрични примитиви.[1] В йерархията на геометричните примитиви, точката и линията представляват фундаментални строителни единици. На тяхна основа се конструират по-сложни двумерни примитиви като окръжности, елипси, полигони, правоъгълници и дъги.[1, 3] В разширени графични библиотеки се включват и тримерни примитиви като куб, сфера и цилиндър. Простите геометрични примитиви служат като структурна основа, върху която се изграждат комплексни изображения и модели. Те представляват базовите елементи, чиито свойства и взаимни отношения позволяват синтеза на сложни визуални репрезентации. В рамките на този литературен обзор ще бъде анализиран алгоритъма на Брезенхам за линия.

II. Алгоритъм на Брезенхам

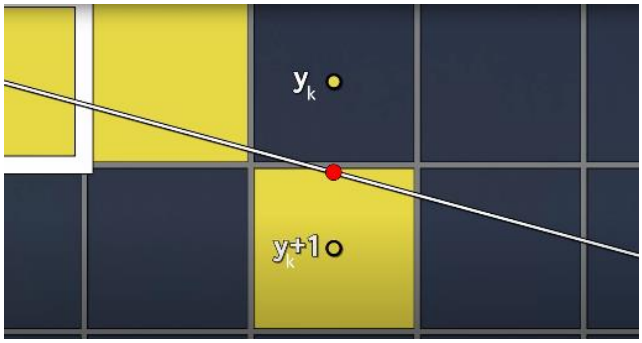
В този раздел ще бъде направен математически обзор на алгоритъма за растерно генериране на линии, разработен от Брезенхам, който използва само инкрементални изчисления с цели числа.

Алгоритъмът на Брезенхам за чертаене на линия може да бъде адаптиран за изчертаване на окръжности и други криви.[1]

Хоризонталните координати на пикселите по линията се определят от дискретни целочислени стойности разделени с еднакви интервали по оста x. Имайки в предвид, че чертаенето на линията започва от лявата крайна точка, преминавайки през колоните по оста x, вертикалната координата на пиксела, която е също целочислено число, взима стойност най-близка до идеалният път на линията.[1, 2] За да се определи положението на правата линия в равнината, се използва следното уравнение:

$$y = mx + b \qquad (1)$$

Нека пикселът (x_k, y_k) е вече поставен на екрана. Хоризонталната координата на следващия пиксел е лесно да се определи защото, това е числото, което веднага върви след x_k, т.е. $x_{k+1} = x_k + 1$. Обаче това не е случай с вертикалната координата, защото стойността изчислена от уравнение (1) може да не е цяло число, и така y_{k+1} може да бъде в интервала на y_k и $y_k + 1$. Алгоритъмът на Брезенхам избира координатата y_k или $y_k + 1$ в зависимост от това, коя от тези две стойности е по-близка до изчислената стойност на y от уравнение (1). [1, 3] С други думи казано, при алгоритъма на Брезенхам, ако е вече поставен един пиксел в равнината на координатите (x_k, y_k), следващият пиксел ще е с координати $(x_k + 1, y_k)$ или $(x_k + 1, y_k + 1)$, в зависимост от това, кое от двете по-точно описва идеалния път на линията.[1] На фиг. 1 са визуално представени точките $(x_k + 1, y_k)$, $(x_k + 1, y_k + 1)$ и идеалният път на линията.



(Фиг. 1) Илюстрация на дискретните стойности на пикселите и идеалният път на линията

Нека разстоянието между y_k и y, изчислено от уравнение (1), бъде означено като d_{lower}. Съответно нека разстоянието между $y_k + 1$ и y бъде означено като d_{upper}. Те могат да се представят по следния начин:

$$d_{lower} = y - y_k = m(x_k + 1) + b - y_k \quad (2)$$

$$d_{upper} = (y_k + 1) - y = y_k + 1 - m(x_k + 1) - b \quad (3)$$

Определянето кой от двата пиксела е по-близо до линията се прави с параметър, който се изчислява от разликата на d_{lower} и d_{upper}. Нека този параметър бъде означен като p_k. Параметърът е необходимо да бъде цяло число и това може да се осъществи със субституцията $m = \frac{\Delta y}{\Delta x}$, където Δx е хоризонталното, а Δy вертикалното разстояние на крайните точки на линията. [1, 3, 4] Параметърът p_k може да се представи по следния начин:

$$p_k = \Delta x(d_{lower} - d_{upper}) \quad (4.1)$$

$$p_k = \Delta x((m(x_k + 1) + b - y_k) - (y_k + 1 - m(x_k + 1) - b)) \quad (4.2)$$

$$p_k = \Delta x((\frac{\Delta y}{\Delta x}(x_k + 1) + b - y_k) - (y_k + 1 - \frac{\Delta y}{\Delta x}(x_k + 1) - b)) \quad (4.3)$$

$$p_k = \Delta x(\frac{\Delta y}{\Delta x}(x_k + 1) + b - y_k) - \Delta x(y_k + 1 - \frac{\Delta y}{\Delta x}(x_k + 1) - b) \quad (4.4)$$

$$p_k = (\Delta y(x_k + 1) + \Delta x b - \Delta x y_k) - (\Delta x y_k + \Delta x - \Delta y(x_k + 1) - \Delta x b) \quad (4.5)$$

$$p_k = \Delta y x_k + \Delta y + \Delta x b - \Delta x y_k - \Delta x y_k - \Delta x + \Delta y x_k + \Delta y + \Delta x b \quad (4.6)$$

$$p_k = 2\Delta y x_k + 2\Delta y + 2\Delta x b - 2\Delta x y_k - \Delta x \quad (4.7)$$

$$p_k = 2\Delta y x_k - 2\Delta x y_k + 2\Delta x b + 2\Delta y - \Delta x \quad (4.8)$$

$$p_k = 2\Delta y x_k - 2\Delta x y_k + \Delta x(2b - 1) + 2\Delta y \quad (4.9)$$

Причината защо Δx се умножава с разликата на d_{lower} и d_{upper} е премахването на Δx под дробната черта, което може да се забележи в уравнение (4.4). Премахването на Δx осигурява параметъра бъде целочислено число.[3] От уравнение (1), b може да се изрази по следния начин:

$$b = y - mx = y - \frac{\Delta y}{\Delta x}x \quad (5)$$

Вмъкването на b от уравнение (5) в уравнение (4.9) се получава най-простата форма на уравнението описващо p_k. Това е показано в уравнение (6.5).[3] В уравненията (6.1), (6.2), (6.3) и (6.4) се използват стойностите на x и y от k-тата стъпка, защото се вече знае, че този пиксел е вече на линията.

$$p_k = 2\Delta y x_k - 2\Delta x y_k + \Delta x(2(y_k - \frac{\Delta y}{\Delta x}x_k) - 1) + 2\Delta y \quad (6.1)$$

$$p_k = 2\Delta y x_k - 2\Delta x y_k + \Delta x(2y_k - 2\frac{\Delta y}{\Delta x}x_k - 1) + 2\Delta y \quad (6.2)$$

$$p_k = 2\Delta y x_k - 2\Delta x y_k + 2\Delta x y_k - 2\Delta y x_k - \Delta x + 2\Delta y \quad (6.3)$$

$$p_k = 2\Delta y(x_k - x_k + 1) - \Delta x(2y_k - 2y_k + 1) \quad (6.4)$$

$$p_k = 2\Delta y - \Delta x \quad (6.5)$$

Знакът на p_k е същият като знака на $d_{lower} - d_{upper}$, тъй като $\Delta x > 0$. Ако пикселът с вертикалната координата y_k е по-близо до линията, отколкото пикселът с вертикалната координата $y_k + 1$, т.е. $d_{lower} < d_{upper}$, тогава параметърът p_k е отрицателен. В този случай избира се пиксела под линията, в противен случай се избира пиксела над линията. Координатите по линията се променят с единични стъпки по хоризонталната и вертикалната ос. Следователно параметърът p_{k+1} може да се изчисли от предходника му p_k. От уравнение (4.9) може да се забележи, че $\Delta x(2b - 1) + 2\Delta y$ не се променя в никакъв случай и целият израз може да се отбележи като c.[1] Получава се:

$$p_k = 2\Delta y x_k - 2\Delta x y_k + c \quad (7)$$

В случая, когато $p_k < 0$, p_{k+1} може да се изведе от уравненията (4.9) и (7).

$$p_{k+1} = 2\Delta y(x_k + 1) - 2\Delta x y_k + c \quad (8.1)$$

$$p_{k+1} = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + c \quad (8.2)$$

$$p_{k+1} - p_k = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + c - 2\Delta y x_k + 2\Delta x y_k - c \quad (8.3)$$

$$p_{k+1} - p_k = 2\Delta y \quad (8.4)$$

$$p_{k+1} = p_k + 2\Delta y \quad (8.5)$$

В случая, когато $p_k \geq 0$, p_{k+1} може да се изведе по следния начин:

$$p_{k+1} = 2\Delta y(x_k + 1) - 2\Delta x(y_k + 1) + c \quad (9.1)$$

$$p_{k+1} = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k - 2\Delta x + c \quad (9.2)$$

$$p_{k+1} - p_k = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k - 2\Delta x + c - 2\Delta y x_k + 2\Delta x y_k - c \quad (9.3)$$

$$p_{k+1} - p_k = 2\Delta y - \Delta x \quad (9.4)$$

$$p_{k+1} = p_k + 2\Delta y - \Delta x \quad (9.5)$$

Параметърът решаващ избора между пиксела над или под линията се определя с рекурсивната релация (10)

$$\begin{cases} p_0 = 2\Delta y - \Delta x \\ p_{k+1} = \begin{cases} p_k + 2\Delta y - \Delta x \text{ ако } p_k \geq 0 \\ p_k + 2\Delta y \text{ ако } p_k < 0 \end{cases} \end{cases} \quad (10)$$

където $k \in N$.

Необходимо е да се обозначи че дефиницията на параметъра p_k от рекурсивната репрезентация (10) добре ще избира пикселите при чертаенето на линия, когато $|m| < 1.0$, защото в противен случай промяната в y е по-бърза от промяната в x, поради което пикселите няма да бъдат поставени в правилните позиции. Решението за този проблем е замяната на ролите на x и y, когато $|m| \geq 1.0$. Координатите на y се увеличават за единица при всяка стъпка, докато параметърът p_k прави решение дали да избере пиксела отляво или отдясно по отношение на идеалния път на линията.

Стъпки на алгоритъма на Брезенхам за $|m| < 1.0$

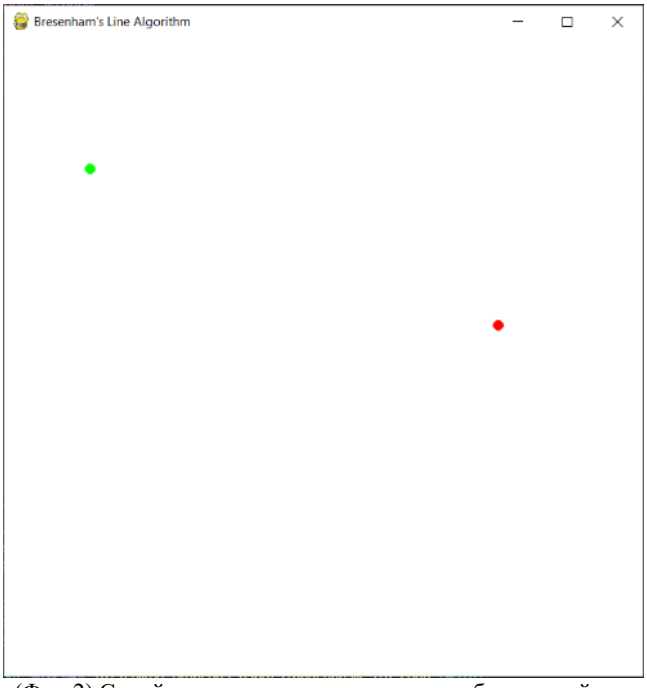
1. Въвеждане на двете точки, които представляват краищата на линията.
2. Изчисляване на Δx, Δy, $2\Delta y$, $2\Delta y - \Delta x$ и p_0. p_0 се изчислява от уравнение (6.5).
3. x се увеличава за единица. p_k се определя от рекурсивното обозначение (10). y се увеличава за едно или остава непроменено, което се определя от p_k.
4. Стъпката 3 се повтаря $\Delta x - 1$ пъти

III. Софтуерна реализация на алгоритъма

Софтуерната реализация има за цел да покаже как описаните математически зависимости могат да бъдат превърнати в конкретни стъпки, които позволяват изчертаването на права линия върху растерна повърхност. В този раздел ще бъде направена софтуерна реализация на алгоритъма на Брезенхам на Python с графичната библиотека Pygame. Библиотеката Pygame е използвана единствено за обработване на събитията на прозореца и за поставяне на отделните пиксели върху екрана, докато основната логика на алгоритъма за изчертаване на линията е реализирана независимо, без използване на вградени графични функции за рисуване. Алгоритъмът е Брезенхам е имплементиран под формата на класова структура. Класа има конструктор и метод за изчисляване на координатите на линията. Класа има член-променлива self.points, който представлява масив от точки описващи линията. Този масив се предава към графичния интерфейс. Методът изчисляващ координатите на точките взима като параметри координатите на началните две точки, и при всяка стъпка добавя нова точка към масива self.points. Кодът на функцията е следният:

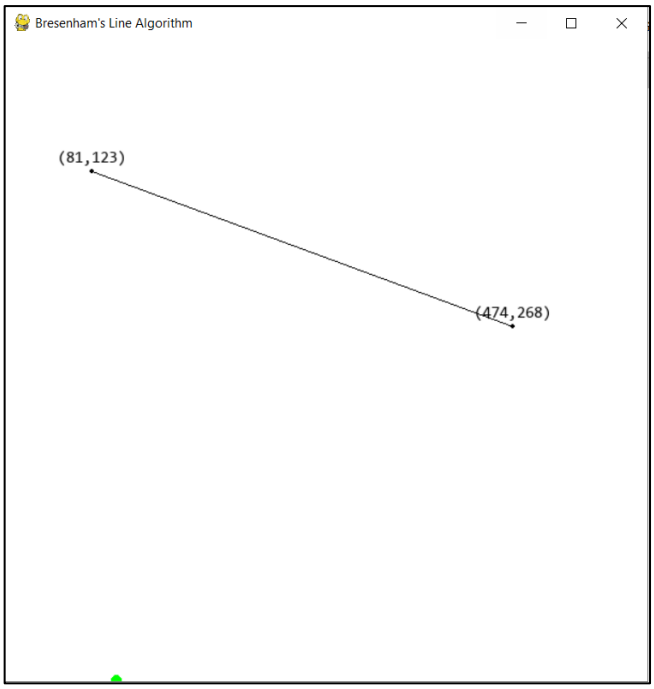
```
def line(self, x1, y1, x2, y2):
    self.points = []

    delta_x = abs(x2 - x1)
    delta_y = abs(y2 - y1)
    step_x = 1 if x2 > x1 else -1
    step_y = 1 if y2 > y1 else -1
    x = x1
    y = y1

    if delta_y < delta_x: # |slope| < 1
        p = 2 * delta_y - delta_x
        for _ in range(delta_x + 1):
            self.points.append((x, y))
            x += step_x
            if p >= 0:
                y += step_y
                p += 2*delta_y - 2*delta_x
            else:
                p += 2 * delta_y

    else: # |slope| >= 1
        p = 2 * delta_x - delta_y
        for _ in range(delta_y + 1):
            self.points.append((x, y))
            y += step_y
            if p >= 0:
                x += step_x
                p += 2 * delta_x - 2 * delta_y
            else:
                p += 2 * delta_x
```

Алгоритъмът разглежда два основни случая в зависимост от наклона на линията. При $|\Delta y| < |\Delta x|$ $(|m| < 1.0)$ параметърът p избира пикселът отгоре или отдолу на идеалният път на линията. При $|\Delta y| \geq |\Delta x|$ $(|m| \geq 1.0)$ параметърът p избира пикселът отдясно или отляво на линията. След приключване на итеративния процес всички координати на определените пиксели се съхраняват в списъка self.points, който представлява крайния резултат на алгоритъма. Той се предава към графичният интерфейс. Необходимо е да се определи посоката, в която алгоритъмът ще се придвижва по координатните оси. Това се осъществява чрез задаването на променливи, обозначени като step_x и step_y. Тяхната стойност (-1 или 1) определя дали при всяка итеративна стъпка координатата ще се увеличава или намалява. По този начин се гарантира, че алгоритъмът може да обработва всички възможни посоки на линията, както с положителен, така и с отрицателен наклон. Не е необходимо отделно разглеждане на всеки квадрант. На фиг. 2 е показано как изглежда избирането на двете крайни точки при чертаенето на линията. При събитието на натискане на левия бутон на мишката се избира началната точка. При събитието на пускане на левия бутон на мишката се избира крайната точка. Съответно, при избор точките се оцветяват в зелено и червено.



(Фиг. 2) Скрийншот на приложението при избор на крайните точки

След избирането на двете крайни точки, имплементираният алгоритъм чертае линията и изписва им координатите. Това е показано на фиг. 3.



(Фиг. 3) Скрийншот на приложението с начертана линия

При събитието на натискане на клавиша SPACE се изтриват линиите на екрана.

Кодът е качен в Github на следния линк: https://github.com/boce1/bresenham_line_CG.git

Заключение

Алгоритъмът на Брезенхам представлява едно от най-ефективните и елегантни решения за изчертаване на права линия в растерна среда. Алгоритъмът използва рекурентна зависимост за вземане на решение коя от двете възможни точки е по-близо до идеалната права. В софтуерната реализация беше показано как математическите зависимости могат да бъдат приложени на практика.

Използвана литература

[1] Arta Moro Sundjaja. (2019). Check Bresenham Algorithm Implementation Analisys in Raster Shape

[2] Hadi Sutopo. (2020). Bresenham's Lines Algorithm Visualization Using Flash. International Journal of Computer Theory and Engineering, Vol. 3, No. 3

[3] Makanjuola Daniel. (2017). A Mathematical Overview of Bresenham Algorithms in the Determination of Active Pixel Positions. (PDF) A Mathematical Overview of Bresenham Algorithms in the Determination of Active Pixel Positions

[4] Bresenham, J. E. (1965). "Algorithm for computer control of a digital plotter". IBM Systems Journal. https://web.archive.org/web/20080528040104/http://www.research.ibm.com/journal/sj/041/ibmsjIVRIC.pdf