



UNIVERSITAT ROVIRA I VIRGILI

## **Exercici 1. Dibuixar una recta**

**Arey Ferrero Ramos**

**5 d'octubre del 2020**

## Qüestions

**Perquè en les implementacions de l'algorisme de Bresenham es fa una consulta sobre el pendent de la recta? Identifica com has fet aquesta consulta en el codi que has utilitzat i quin significat i raó de ser li dones.**

En les implementacions de l'algorisme de Bresenham es fa una consulta sobre el pendent de la recta perquè el valor del pendent que tingui la recta serà el que determinarà si en cada iteració del bucle encarregat del traçat de la recta el següent píxel que s'hagi de pintar s'obtindrà variant els valors de les dues coordenades (avenç inclinat) o únicament variant el valor d'una de les coordenades (avenç recte).

Aquesta consulta es fa diverses vegades en el codi. Primer de tot, es fa en l'apartat encarregat de calcular els increments per a les seccions amb avenç inclinat, a on es fan els càlculs pertinents en relació a si el pendent es positiu o negatiu respecte a l'eix de les X i respecte a l'eix de les Y. Després es fa una segona consulta en l'apartat encarregat de calcular els increments per a les seccions amb avenç recte, a on es fan els càlculs pertinents en relació a si el pendent és inferior o superior a 1. Finalment, en l'apartat d'inicialització de les variables restants i a partir de la distancia entre els dos punts respecte a cada un dels eixos es fa un càlcul relatiu al valor del pendent, que és el que s'utilitzarà per determinar si les coordenades del següent píxel a pintar s'obtindran seguint un avenç inclinat o seguint un avenç recte.

**Si ens demanen de pintar una recta vertical, podríem optimitzar l'algorisme d'alguna forma? Com ho faries? Ho té en compte el codi que has fet?**

L'algorisme es podria optimitzar per a pintar una recta vertical.

Per fer-ho, s'ha de tenir en compte que en una recta vertical, el valor de x1 y el valor de x2 és el mateix. Això implica que, en el bucle encarregat de pintar aquesta recta, només es variarà el valor de la coordenada y. En conseqüència, per implementar aquesta optimització, s'hauria d'afegir un condicional que, en cas que el valor de x1 y el valor de x2 fossin iguals, en el cos d'aquest es pintessin tots els píxels que componen la recta sense haver de fer cap càlcul per comprovar si s'ha de variar el valor de x (avenç inclinat) o no (avenç recte), ja que el valor de x mai variarà.

El codi que he fet inclou l'optimització de l'algorisme per a pintar una recta vertical y també per pintar una recta . Seguidament s'inclou el pseudocodi d'aquestes optimitzacions:

```
si (x1 = x2) llavors  
    per (y := y1; y < y2, y := y + 1) fer  
        dibuixar_punt(x1, y);
```

```

        fper
    fsi

    si (y1 = y2) llavors
        per (x := x1; x < x2; x := x + 1) fer
            dibuixar_punt(x, y1);
        fper
    fsi

```

### **Les mateixes preguntes si ens demanen de pintar una recta de pendent 1.**

L'algorisme es podria optimitzar per a pintar una recta de pendent 1.

Per fer-ho, s'ha de tenir en compte que en una recta de pendent 1, el valor absolut de  $y_2 - y_1$  y el valor absolut de  $x_2 - x_1$  és el mateix. Això implica que, en el bucle encarregat de pintar aquesta recta, sempre es variaran els valors de les dues coordenades alhora. Dit d'una altra manera, no hi haurà cap cas en el que es variï el valor d'una coordenada i no es variï el valor de l'altra. En conseqüència, per implementar aquesta optimització, s'hauria d'afegir un condicional addicional que, en cas que el valor absolut de  $y_2 - y_1$  y el valor absolut de  $x_2 - x_1$  fossin iguals, en el seu cos es pintessin tots els píxels que componen la recta sense haver de fer cap càlcul per comprovar si no s'ha de variar el valor de la coordenada x o no s'ha de variar el valor de la coordenada y, ja que mai es variarà el valor d'una de les coordenades sense variar el valor de l'altre.

El codi que he fet inclou l'optimització de l'algorisme per a pintar una recta de pendent 1. Per mantenir l'estructura de l'algorisme de Bresenham, he utilitzat disseny descendent. Seguidament s'inclou el pseudocodi d'aquesta optimització:

```

    si (absolut(y2 - y1) = absolut(x2 - x1)) llavors
        dibuixar_recta_mitja(x1, y1, x2, y2);
    fsi

acció dibuixar_recta_mitja(x1: enter, y1: enter, x2: enter, y2: enter) és
var
    x, y, xi, yi: enter;
fvar
inici
    si (x2 > x1) llavors
        xi = 1;

```

```

sino
    xi = -1;

fsi
si (y2 > y1) llavors
    yi = 1;

sino
    yi = -1;

fsi
y := y1;

per (x := x1; x <> x2; x := x + xi) fer
    dibuixar_punt(x, y);
    y := y + yi;

fper
faccio

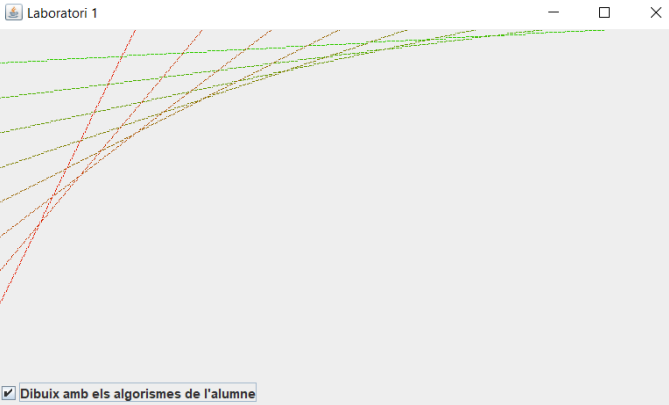



```


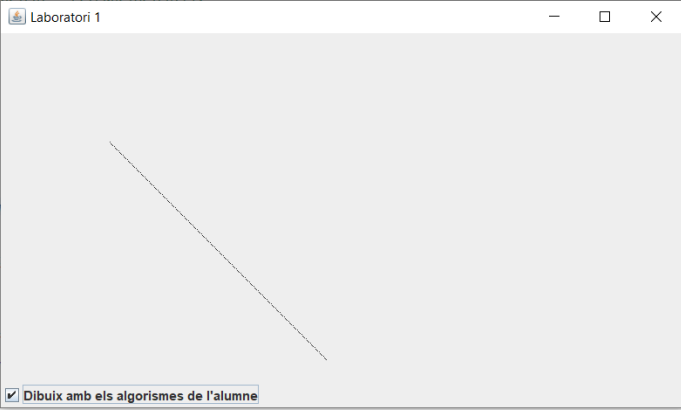


**Quina és la principal diferència entre l'algorisme DDA i el Bresenham? Quin efecte té en els casos particulars de les dos preguntes anteriors?**



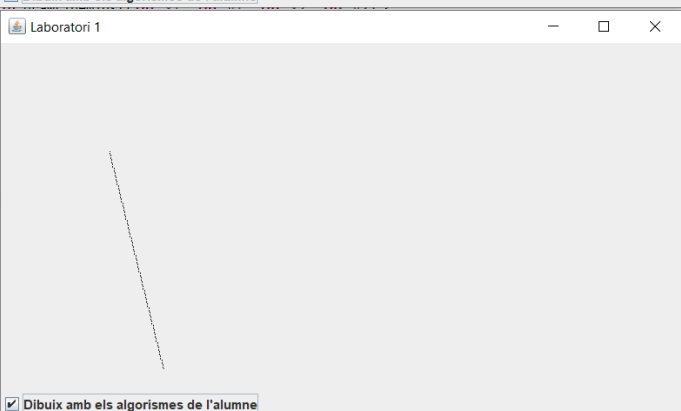
La principal diferencia entre l'algorisme DDA i l'algorisme de Bresenham és que l'algorisme DDA utilitza valors reals (en coma fixa o en coma flotant) per fer els càlculs aritmètics mentre que l'algorisme de Bresenham utilitza valors enters. A més, les operacions utilitzades en l'algorisme DDA són multiplicacions i divisions mentre que les operacions utilitzades en l'algorisme de Bresenham són fonamentalment sumes i restes. Tot això implica una reducció significativa del temps de càlcul de l'algorisme de Bresenham respecte a l'algorisme DDA i, per tant, l'algorisme de Bresenham és molt més ràpid que l'algorisme DDA. A més, tot i que d'entrada pugui semblar el contrari ja que l'algorisme DDA utilitza valors reals, l'algorisme de Bresenham és més precís que l'algorisme DDA, ja que en aquest últim es perd informació durant el procés d'arrodoniment dels càlculs intermedis. Així, es pot afirmar que l'algorisme de Bresenham és més òptim que l'algorisme DDA en termes d'usabilitat, cost i us.

Per tot el mencionat anteriorment, l'algorisme de Bresenham pintarà una recta vertical i una recta de pendent 1 en un temps inferior al que trigarà en pintar-la l'algorisme DDA i, en el cas de que les rectes siguin molt llargues, l'algorisme de Bresenham també les pintarà de forma més precisa que l'algorisme DDA. Si, a més s'afegeixen les optimitzacions dissenyades, l'algorisme de Bresenham serà encara més ràpid del que ja era i, per tant, serà molt més ràpid que l'algorisme DDA.

## Joc de proves



Prova	Descripció	Correcte?
1	Prova de fum.	
2	Recta vertical on els vèrtexs s'indiquen de dalt a baix. <code>dibuixar_recta(10 0, 100, 100, 300)</code>	
3	Recta vertical on els vèrtexs s'indiquen de baix a dalt. <code>dibuixar_recta(10 0, 300, 100, 100)</code>	
4	Recta horitzontal on els vèrtexs s'indiquen d'esquerra a dreta. <code>dibuixar_recta(10 0, 100, 300, 100)</code>	

<p><b>5</b></p>	<p>Recta horitzontal on els vèrtexs s'indiquen de dreta a esquerra.</p> <pre>dibuixar_recta(300, 100, 100, 100)</pre>	
<p><b>6</b></p>	<p>Recta amb pendent = 1 on els vèrtexs s'indiquen d'esquerra a dreta.</p> <pre>dibuixar_recta(100, 100, 300, 300)</pre>	
<p><b>7</b></p>	<p>Recta amb pendent = 1 on els vèrtexs s'indiquen de dreta a esquerra.</p> <pre>dibuixar_recta(300, 300, 100, 100)</pre>	
<p><b>8</b></p>	<p>Recta amb pendent &lt; 1 on els vèrtexs s'indiquen d'esquerra a dreta.</p> <pre>dibuixar_recta(100, 100, 300, 150)</pre>	

9	<p>Recta amb pendent &lt; 1 on els vèrtexs s'indiquen de dreta a esquerra.</p> <pre>dibuixar_recta(300, 150, 100, 100)</pre>	
10	<p>Recta amb pendent &gt; 1 on els vèrtexs s'indiquen d'esquerra a dreta.</p> <pre>dibuixar_recta(100, 100, 150, 300)</pre>	
11	<p>Recta amb pendent &gt; 1 on els vèrtexs s'indiquen de dreta a esquerra.</p> <pre>dibuixar_recta(150, 300, 100, 100)</pre>	

### Consideres que falta algun cas per validar?

Sí. Considero que també s'hauria de validar al menys dos cassos més: El cas en que s'ha de dibuixar una recta de pendent -1 on els vèrtexs s'indiquen d'esquerra a dreta i el cas en que s'ha de dibuixar una recta de pendent -1 on els vèrtexs s'indiquen de dreta a esquerra.

Prova	Descripció	Correcte?
1	Recta amb pendent = -1 on els vèrtexs s'indiquen de dreta a esquerra. <pre>dibuixar_recta(100, 300, 300, 100)</pre>	
2	Recta amb pendent = -1 on els vèrtexs s'indiquen de dreta a esquerra. <pre>dibuixar_recta(300, 100, 100, 300)</pre>	

### Calia considerar tots aquests cassos? Raona la resposta

Sí calia considerar tots aquests cassos per diversos motius. Primer de tot, els cassos en els que es dibuixa una recta vertical, una recta horitzontal o una recta de pendent 1, son implementacions addicionals realitzades pel propi alumne per optimitzar el codi. Només per aquest motiu, ja s'haurien de fer totes les comprovacions necessàries per garantir que aquests cassos funcionen de forma correcta. També s'ha de considerar aquests tres cassos perquè es tracta de cassos límits a on és molt més probable que el programa falli. Addicionalment, s'ha de comprovar els cassos en que es dibuixa una recta en el que el pendent és superior a 1 i en el que el pendent és inferior a 1 perquè en un cas la distància entre  $x_1$  i  $x_2$  és superior a la distància entre  $y_1$  i  $y_2$  mentre que en l'altre cas es produeix el fenomen invers.

Com a exemple de tot el que s'ha mencionat anteriorment, l'algorisme de Bresenham que està penjat a la Wikipedia en castellà no funciona de manera correcta per als cassos en que la distancia entre  $x_1$  i  $x_2$  és inferior a la distancia entre  $y_1$  i  $y_2$ , ja que no dibuixa els píxels en els que el valor de la coordenada  $x$  (que varia de  $x_1$  fins a  $x_2$ ) és igual a  $x_2$  mentre que l'algorisme que està penjat a la Wikipedia en anglès sí que funciona per aquests cassos. Això es degut a que el primer algorisme no es fan totes les comprovacions prèvies necessàries relatives al bucle principal (`mentre (x = x2) fer`). La existència d'aquest error és fàcil de detectar amb un joc de proves en el que es comprovin els cassos límits, ja que no es dibuixarà la recta vertical. Amb una anàlisi



posterior en profunditat es podrà observar que tampoc funciona per a tots els cassos ja mencionats, facilitant així el descobriment del motiu de l'error.

A més, per cada tipus de recta s'ha de comprovar si es pinta indicant els vèrtexs en un sentit o un altre ja que els valors dels vèrtexs inicials ( $x_1$  i  $y_1$ ) poden ser inferiors, superiors o iguals als valors dels vèrtexs finals ( $x_2$  i  $y_2$ ) i, per cada un d'aquests cassos s'han de fer unes consideracions prèvies diferents en les quals es poden produir errors que no tenen perquè afectar ni al cas en el que els vèrtexs es pinten en sentit oposat ni a cap altre cas.

Evidentment, aquest raonament no té en compte que alguns dels últims quatre cassos ja s'haurien validat amb la prova de fum.