German Telmo Eizaguirre Suarez

Doble Grado en Biotecnología e Ingeniería Informática

27-09-2019

Ejercicio 1. Dibujar una recta

Modelización y Visualización, URV (Tarragona)

Preguntas

¿Por qué en las implementaciones del algoritmo de Bresenham se consulta la pendiente de la recta? Identifica cómo has hecho esta consulta en el código que has utilizado y qué significado y razón de ser le das.

La pendiente de la recta permite determinar si la recta está más cerca de la abscisa de las x o de las y. Esto determina cómo extenderá el código la recta. En el algoritmo, las coordenadas de la abscisa predominante se incrementan de uno en uno a cada iteración mientras que las de la abscisa restante se incrementan en base a un valor de error. El valor de error surge de lo cerca que está la siguiente coordenada teórica de cada uno de los píxeles adyacentes.

En el código, realizo esta consulta calculando la pendiente de la recta de manera estándar ; $\Delta y / \Delta x$. Dependiendo de si su valor es mayor o menor a 1 (45º respecto a ambas abscisas) asocio la recta a un u otro eje.

¿Si nos pidieran dibujar una recta vertical, podríamos optimizar el algoritmo de alguna manera? ¿Cómo lo harías? ¿Lo tiene en cuenta el código que has hecho?

Si la recta fuera totalmente vertical se cumpliría que $x1 = x2 \rightarrow \Delta x = 0$. En primer lugar, habría que establecer un control para evitar la división por 0 en el cálculo de la pendiente. En segundo lugar, podríamos optimizar el código haciendo que, en caso de darse la condición anterior, solo se incrementaran las coordenadas y las x se mantuvieran constantes. De este modo nos ahorraríamos el cálculo del valor de error a cada iteración, ya que nunca habría que incrementar las x. Dicha funcionalidad está incluida en el código. En lugar de controlar que no se divida por cero, en mi caso solamente se calcula la pendiente si $\Delta x \neq 0$. También he adaptado la funcionalidad para rectas horizontales.

Las mismas preguntas si nos pidieran pintar una recta de pendiente 1.

En el caso de una recta de pendiente 1, la aportación de ambas abscisas sería igual. Por lo tanto a cada píxel se incrementaría tanto el valor de las x como el de las y. Con una comprobación más podemos optimizar el código. En mi código, he incluido que si la pendiente es igual a 1, a cada iteración se incrementan ambas coordenadas hasta llegar al punto final, evitando de nuevo el cálculo iterativo del valor de error.

German Telmo Eizaguirre Suarez

Doble Grado en Biotecnología e Ingeniería Informática

27-09-2019

¿Cuál es la principal diferencia entre el algoritmo DDA y el de Bresenham? ¿Qué efecto tiene en los casos particulares de las dos preguntas anteriores?

La principal diferencia entre ambos algoritmos es que el de DDA se basa en multiplicaciones, divisiones y sumas de números de coma flotante, y el de Bresenham en sumas y restas de números enteros. El algoritmo de Bresenham resulta más eficiente y más preciso, aunque más complejo en sus cálculos.

Para los casos anteriores, el algoritmo DDA no requiere una optimización como la que he implementado en el Bresenham. Sin embargo, los números con los que trabaja seguirán siendo de tipo coma flotante, aunque los cálculos para líneas verticales, horizontales y de pendiente 1 generen números enteros. Por lo tanto, DDA resulta menos eficiente que el Bresenham optimizado.

Juego de pruebas

Para el juego de pruebas, he considerado que cuando se indica “de izquierda a derecha” o viceversa, o “de arriba a abajo” o viceversa, es referente a la posición en la pantalla. Debemos recordar que la coordenada 0 de las y se sitúa en la cota superior de la pantalla.

Considero que faltan una serie de casos a validar en el enunciado de la práctica. Los he incluido al final del juego de pruebas. Debería poder dibujarse una recta de un solo punto (coordenadas iniciales y finales iguales). Además se deberían incluir rectas con pendientes negativas. En el algoritmo se tratan igual que las de pendiente positiva (únicamente se modifica el valor de incremento de las coordenadas según el tipo de pendiente), sin embargo no está de más probar que funcionan correctamente.

Aparte de los casos evaluables, existen casos no evaluables en este juego de pruebas. Esto se debe a que el entorno de dibujo solo incluye el primer cuadrante de los 4 que forman ambas abscisas. Más concretamente, solo incluye el cuadrante de coordenadas positivas. Deberíamos evaluar también el funcionamiento del algoritmo con coordenadas negativas. Sin recurrir a elementos gráficos, podríamos sustituir drawPoint por la impresión numérica de las coordenadas por la salida estándar.

No está fuera de lugar considerar todos estos casos ya que permiten un coverage completo del código. Aunque algunos puedan ser triviales, nos permiten verificar que las diferentes condiciones y caminos de ejecución se cumplen correctamente.

German Telmo Eizaguirre Suarez

Doble Grado en Biotecnología e Ingeniería Informática

27-09-2019

Caso	**Color**	**Coordenadas**			
		x1	**y1**	**x2**	**y2**
Recta vertical, vértices de arriba a abajo.	Rojo. <1,0,0>	625	4	625	325
Recta vertical, vértices de abajo a arriba.	Rojo oscuro. <0.5, 0, 0>	635	280	635	0
Recta horizontal, vértices de izquierda a derecha.	Verde. <0,1,0>	500	300	550	300
Recta horizontal, vértices de derecha a izquierda.	Verde oscuro. <0, 0.5, 0>	600	280	500	280
Recta de pendiente = 1, vértices de izquierda a derecha.	Naranja. <1, 0.5, 0>	450	300	580	170
Recta de pendiente = 1, vértices de derecha a izquierda.	Marrón oscuro. <0.5, 0.25, 0>	515	120	400	235
Recta de pendiente < 1, vértices de izquierda a derecha.	Cyan a rosa. <0.5, 0.5, 1>	400	235	600	120
Recta de pendiente < 1, vértices de derecha a izquierda.	Azul a cyan. <0, 0.25, 0.5>	580	170	400	300
Recta de pendiente > 1, vértices de izquierda a derecha.	Amarillo. <0.9, 0.9, 0>	400	235	540	0
Recta de pendiente > 1, vértices de derecha a izquierda.	Amarillo oscuro. <0.7, 0.7, 0>	450	327	580	170
Punto único.	Fucsia. <0.941, 0.071, 0.745>	400	100	400	100
Recta de pendiente < 0, vértices de izquierda a derecha.	Morado. <0.886, 0.156, 0.749>	300	70	420	150
Recta de pendiente < 0, vértices de derecha a izquierda.	Morado oscuro. <145, 89, 135>	420	190	280	90