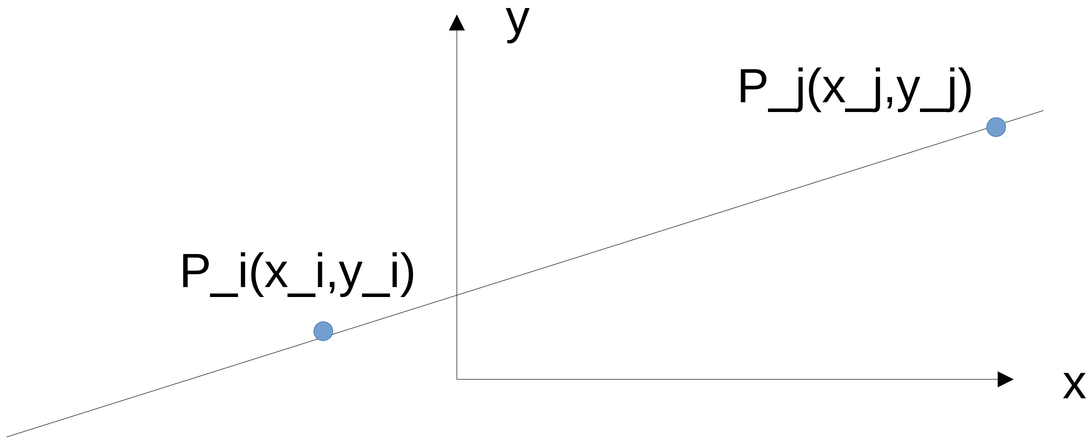
Introduction to 2D Vector Graphics



Note:
(1) for alpha = 0 and 1 equation (2) returns P(x,y) = P_i(x_i, y_i), P(x,y) = P_j (x_j , y_j);

Based on the above vector form equation, generate screen savers and trees. (Reference: H. Li, IEEE Transactions on Education)

Direction vector d(x,y) = (dx, dy), which is defined as

d(x,y) = P_i(x_i,y_i) – P_j(x_j,y_j) … (1)

= (x_i – x_j, y_i – y_j) … (1-1)

Now the vector form equation for the straight line:

P(x,y) = P_i(x_i,y_i) + alpha * (P_i(x_i,y_i) – P_j(x_j,y_j))

… (2)

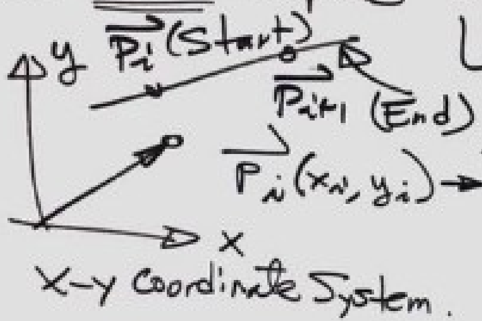
Where 0 <= alpha <= 1

Harry Li, Ph.D.

9-17-2018 2D Vector Algorithm Example

CMPE240 Adv. Microprocessor Systems. Sept. 17, 2018 HL .3/.

2D Vector Graphics V.S. Pixel graphics



Line Segment. $\boxed{\vec{P}(x,y) \overset{?}{=} \vec{P_i} + \lambda \vec{d} \quad \cdots (1)}$

where $\boxed{\vec{d} \triangleq \vec{P_{i+1}} - \vec{P_i} \quad \cdots (2)}$

C/C++ To Compute Eqn (2).

physical meaning of Eqn (1):

Question 1: To Get "Start" pt on the Line Segment, what is the $\lambda = ?$ $\lambda = 0$

To the "Ending" pt, $\lambda = ?$ $\lambda = 1$.

all the pts B/W $\vec{P_i}$ and $\vec{P_{i+1}}$, $\lambda = ?$ $0 < \lambda < 1$

all the pts Beyond $\vec{P_{i+1}}$, $\lambda = ?$ $\lambda > 1$

" pts .. Below $\vec{P_i}$, $\lambda = ?$ $\lambda < 0$

Example: Given $\vec{P_i}$, and $\vec{P_{i+1}}$, find A direction vector $\vec{d}$.

Sol from Eqn (2), we have

$$\vec{d} = \vec{P_{i+1}} - \vec{P_i}$$

$$(x_d, y_d) = (x_{i+1}, y_{i+1}) - (x_i, y_i) = (x_{i+1} - x_i, y_{i+1} - y_i)$$

X_d = x[i+1] - x[i];

y_d = y[i+1] - y[i];

9-17-2018 2D Vector Rotating Sqrs

Example: Implement 2D Vector Graphics On S.P.I. LCD Display.

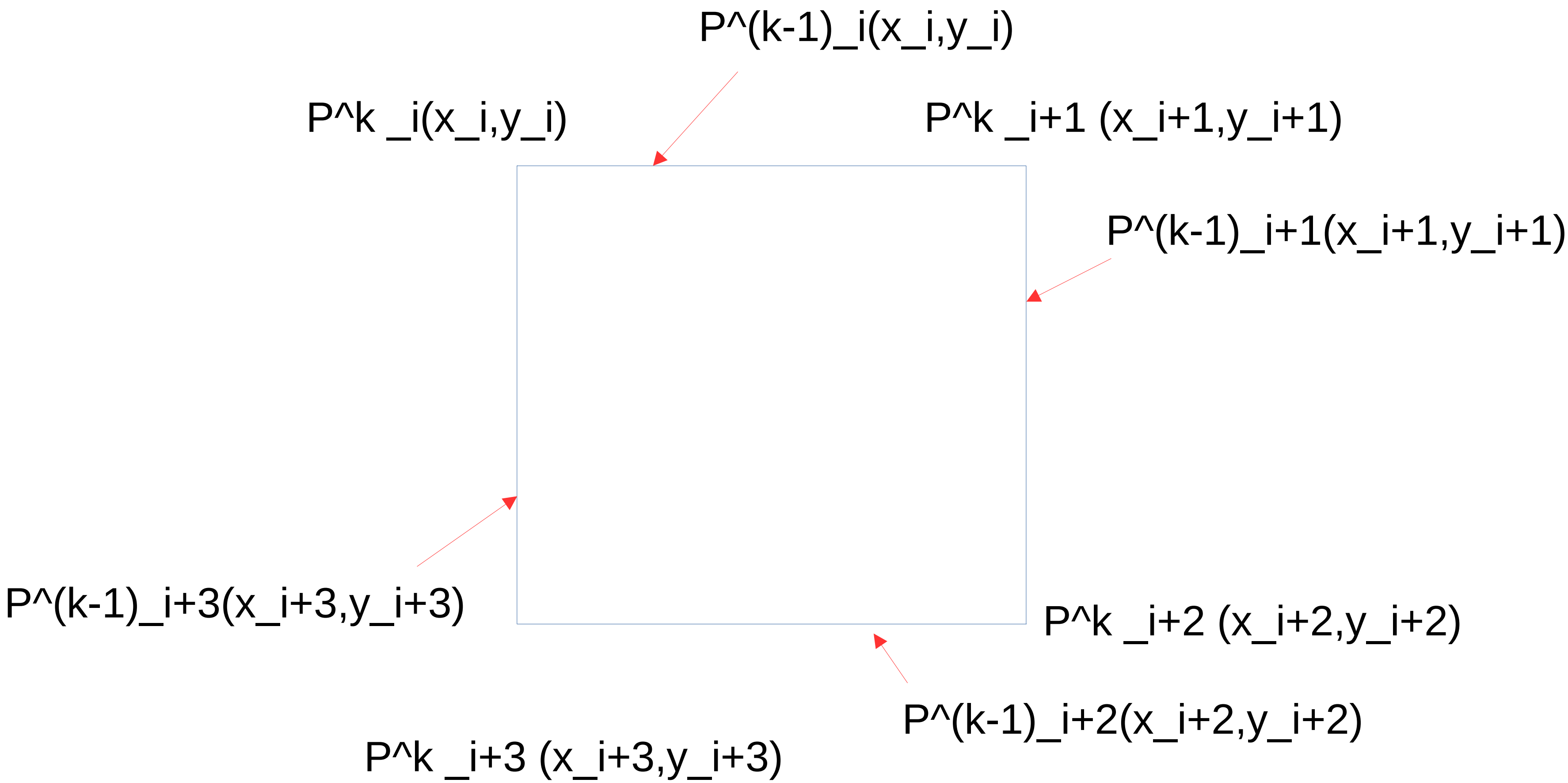
Sol $\{\overrightarrow{P_i}(x_i, y_i) \mid i = 0, 1, 2, 3\}$

From Eq(1), we have

$$\overrightarrow{P_i^{j+1}} = \overrightarrow{P_i} + \lambda(\overrightarrow{P_{i+1}^{j}} - \overrightarrow{P_i^{j}})$$

Let $\lambda = 0.2$

2D Vector Graphics to Create Rotating Pattern



From equation (1), we can derive the following equation

$$P^k (x,y) = P^{(k-1)}_i(x_i,y_i) + \text{alpha} * (P^{(k-1)}_i(x_i,y_i) - P^{(k-1)}_j(x_j,y_j)) \quad \ldots (3)$$

Choose alpha = 0.8

Harry Li, Ph.D.

2D Rotating Pattern Technique

I have created an algorithm to summarize the rotating pattern technique:

Defining a polygon with a set of vertices $\{(x_i, y_i)|i = 1, 2, \cdots, k\}$, one can use a vector formula to describe an object reduction and rotation as:

$$(x_i^{l+1}, y_i^{l+1}) = (x_i^l, y_i^l) + \mu(x_{i+1}^l - x_i^l, y_{i+1}^l - y_i^l) \qquad (1)$$

where the subscript i is used to denote each vertex of a given object, $i = 1, 2, \cdots, k$. If $i = k$ and $i + 1 > k$, then $i + 1 = 1$. The superscript l is used to denote the level of iteration. The constant μ is defined as $0 \leq \mu \leq 1$, which is used to define the rate of reduction. For example, if $\mu = 0.5$ then each side of the object is reduced to half. This μ is also related to the direction of the rotation. For μ less than 0.5, the rotation is toward the current reference point (x_i^l, y_i^l), otherwise the rotation is away from the current point. Equation (1) is, in fact, derived directly from a vector addition. Let us assume $\overline{a} = (x_i^l, y_i^l)$, $\overline{b} = (x_{i+1}^l - x_i^l, y_{i+1}^l - y_i^l)$, and $\overline{a}' = (x_i^{l+1}, y_i^{l+1})$, then (1) can be written as $\overline{a}' = \overline{a} + \mu(\overline{b} - \overline{a})$ for vertices i and $i + 1$ with reduction at level l. Following the argument above, μ can also be expanded to the range greater than 1 to produce magnification.

Reference:

IEEE TRANSACTIONS ON EDUCATION, VOL. 35, NO. 1,
Three-Dimensional Computer Graphics Using EGA or VGA Card By H. Li