Formule pour dévier la position d’un curseur

Arnaud Beauny, Santiago Muñoz

x0, y0 est la position initiale de la souris (point de départ). Ce point de départ reste fixe pour la durée de l’essai: toutes les nouvelles positions seront basées sur ce point initial. x0, y0 ne doit donc pas être mis à jour pendant le mouvement.

x1, y1 est la nouvelle position de la souris. x1 et y1 sont mis à jour à chaque rafraichissement de l’écran, par exemple.

x2, y2 sont les valeurs obtenues après avoir appliqué la formule de déviation.

N.B. Updating both y AND x to keep the same speed on both axis. If I only updated y, deviant movements would go faster than non deviated.

**Matlab:**

For ‘a’ the angle amplitude in degrees:

x2 = sqrt((x1-x0)^2+(y1-y0)^2) \* cosd(a + atan2((y1-y0),(x1-x0))\*180/pi) + x0

y2 = sqrt((x1-x0)^2+(y1-y0)^2) \* sind(a + atan2((y1-y0),(x1-x0))\*180/pi) + y0

**Python (Numpy):**

For ‘x’ the angle amplitude in degrees:

*# Convert degree angle to radiants*

# a = x \* pi / 180 # No pi in python 2.7

a = np.deg2rad(x)

*# Formules*

x2 = np.sqrt((x1-x0)\*\*2+(y1-y0)\*\*2) \* np.cos(a + np.arctan2((y1-y0),(x1-x0))) + x0

y2 = np.sqrt((x1-x0)\*\*2+(y1-y0)\*\*2) \* np.sin(a + np.arctan2((y1-y0),(x1-x0))) + y0

N.B. if units are pixels, use round() to round the result (half pixels don’t exist)