Application of Affine Transformations in a Mathematical Cartesian Coordinates for Java Students

Hieu Vu
Computer Science Department
American University of Nigeria
Yola, Adamawa State, Nigeria
E-mail: hieu.vu52@yahoo.com.

ABSTRACT

This paper presents affine transformations of objects in Java programming language in a conventional coordinate system. Affine transformation preserves the original shape of an object; therefore it is a very important aspect in computer graphics. This paper presents a technique, how to apply affine transformation in a mathematical Cartesian coordinates for Java students.

Keywords: *Affine transformation,* Distortion, *Java programming language*

INTRODUCTION

In computer science, the official language of graphics is OpenGL (Open Graphics Library). It is a component of C++ programming language for graphics. For the last 10 years, Java has replaced C++ as the dominant language for teaching computer programming in most universities and colleges in the United States and across the world. How can a student without background in C++ learn computer graphics? There are two types of transformations of objects (Distortion and Affine). Distortion can be achieved by moving the anchor points and control points, distortion transformations will change the original shape of objects. Affine transformation is applied to preserve the fundamental or original shape of objects. The most important of affine transformations are: translation, scaling, rotation about a point, reflection about a line and shearing (distortion of an angle). This paper will present a method on how to apply affine transformations in a mathematical Cartesian coordinates for Java students. The Java coordinate system places the origin (0, 0) at the upper left corner, stretches out to the right as x-coordinate and downward as the y-coordinate. The points (pixels) reside inside the quadrant (screen), all have positive coordinates (Figure 1). In a conventional coordinate system, the origin (0, 0) should be at the center of the display area and the two axes (x-coordinate, y-coordinate) divide the display area into four quadrants (Figure 1).

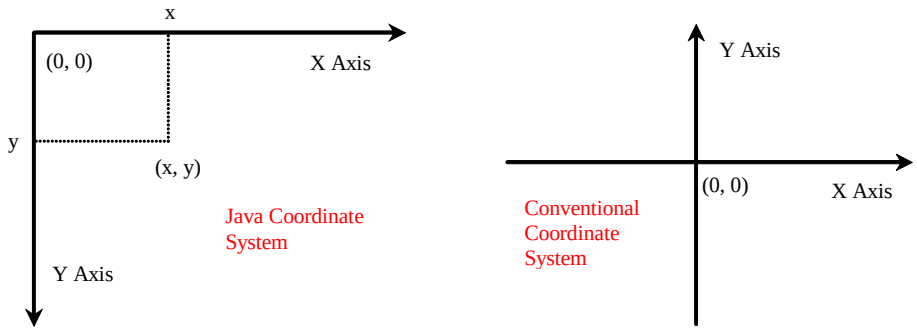

Figure 1: Java and conventional (Cartesian) coordinate systems (Liang, 2009)

Conversion of Java coordinate system to conventional coordinate system

The origin of the conventional system is set at the center of the display area. For an example, the display area is typically 640 x 480 pixels.

```
JFrame frame = new JFrame("TRANSLATION"); //Create frame
frame.getContentPane().add(new Translation());
frame.setSize(640, 480);        //Set frame size
frame.setVisible(true);
```

Therefore, the origin of the conventional coordinate system should have coordinates:

$$x = \frac{screenWidth}{2} = \frac{640}{2} = 320$$

$$y = \frac{screenHeight}{2} = \frac{480}{2} = 240 \quad (1)$$

We now have the origin of the conventional coordinate system which is set at O (320, 240).

Conversion of points into conventional coordinate system

First, we have divided the x and y axes of the conventional coordinate system into halves, therefore the x and y-coordinates of the points set on the Java display area should also be divided by 2 to fit on the new system. For example, point $P_0(x_0, y_0)$ will have coordinates $({}^{x_0}/_2, {}^{y_0}/_2)$ on the conventional coordinate system. Second, all points on the original Java display area have positive values for both *x* and *y* coordinates and they should appear on the first quadrant of the conventional coordinate system. We have to reset the points with respect to the new origin O (320, 240) of the new coordinate system.

Illustration

First, we select four points on the original screen (display area) then draw the pyramid with the base as a triangle (Figure 2).

```
Point p1 = new Point (100, 100);   //100  100
Point p2 = new Point (260, 200);   //260  200
Point p3 = new Point (220, 300);   //220  300
Point p4 = new Point (60, 260);    //60   260
```

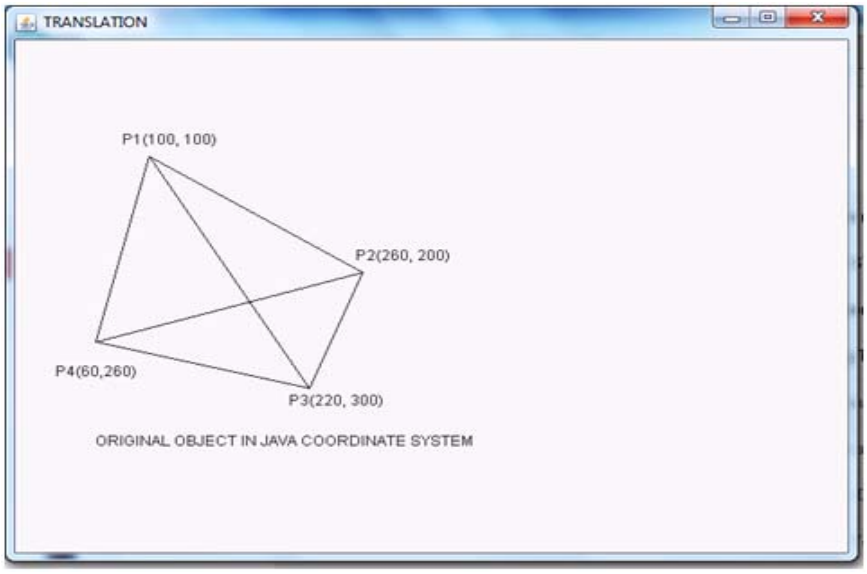

Figure 2: Original shape on screen (Java coordinate system)

Next step:
We find the origin for the conventional coordinate system (1), draw two axes for x and y-coordinates.
Let P'_i be the new points of P_i,$(1 <= i <= 4)$ in the new conventional coordinate system. Since, we divided the *x* and *y*-coordinates of the screen by 2, the coordinates of the new points P'_1 would be halves of the original coordinates.

$$P'_i.x = P_i.x/2 \quad \text{and} \quad P'_i.y = P_i.y/2 \qquad (2)$$

And all the points should be on the first quadrant with respect to conventional origin O (x, y) (both $x_i, y_i > 0$), we have:

$$P'_i.x = P_i.x/2 + O.x$$
$$P'_i.y = P_i.y/2 + O.y \qquad (3)$$

The x-coordinates $P'_i.x$ is in correct positions, to get the y-coordinates $P'_i.y$ in the first quadrant, we need to reflect them through the x-axis of the conventional coordinate system (x = O.y), therefore:

$$(3) \Rightarrow P'_i.x = P_i.x/2 + O.x$$
$$P'_i.y = O.y - P_i.x/2 \qquad (4)$$

The four new points will have coordinates in the conventional system as:

Point p'_1(370, 190)
Point p'_2(450, 140)
Point p'_3(430, 90)
Point p'_4(350, 110)

Then draw the new pyramid on the conventional coordinate system (Figure 3).

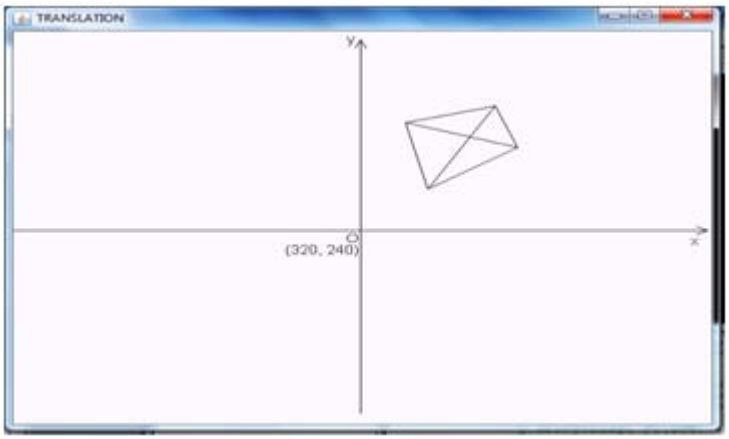

Figure 3: New shape on conventional system

Affine transformations: Affine transformation preserves original shape of objects, it is an important part in computer graphics and has many applications in movie industry, animation, CAD/CAAD, simulation, etc. Affine transformations are linear transformations which include translation, scaling, rotation, reflection, and shearing (Renka, 2013). This paper uses translation transformation with respect to the conventional coordinate system to illustrate an example.

Translation

Translation is the simplest operation of transformation. Translation will move the object to a new position along the x-axis (x pixels) and y-axis (y-pixels). For example, supposed we want to translate the shape in figure 2 with respect to the origin O(320, 240) of the conventional coordinate system a distance 100 pixels along the x-axis and 70 pixels along the y-axis. The translation matrix would be:

$$\begin{matrix} 1 & 0 & 100 \\ 0 & 1 & 70 \\ 0 & 0 & 1 \end{matrix}$$

The point matrix of the shape would be:

$$\begin{matrix} 370 & 450 & 430 & 350 \\ 190 & 140 & 90 & 110 \\ 1 & 1 & 1 & 1 \end{matrix}$$

The result matrix (new point matrix) would be:

$$\begin{matrix} 480 & 550 & 430 & 450 \\ 260 & 210 & 160 & 180 \\ 1 & 1 & 1 & 1 \end{matrix}$$

After performing the translation, draw new picture that contains both original and new shapes (Figure 4).

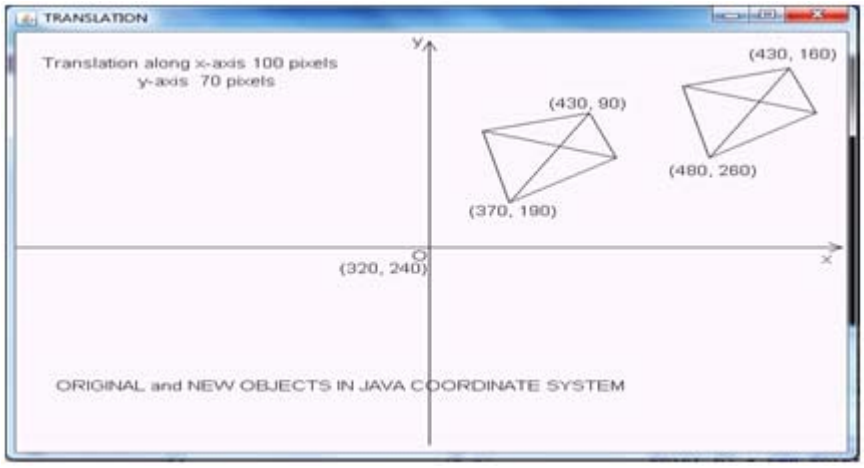

Figure 4: Translation

CONCLUSION

This paper can be used as a guide for graphics programming in Java. As can be seen through the example above, the translation process for an object in a conventional coordinate system can be used in Java. However, to make this happened, we first need to convert the Java coordinate system to conventional system and also the coordinates of points of the original object to their relative coordinates in the new conventional system before applying the translation process. Other transformations such as scaling, rotation, refection, and shearing can be done in the same way.

REFERENCES

Liang (2009). Introduction to Java Programming. Seventh edition. Pearson Education. New Jersey: Upper Saddle River. ISBN-13: 978-0-13-814626-9

Renka, R. J.(2013) Department of Computer Science & Engineering, University of North Texas, 9/23/2013